Getting Started with Model*Sim* PE

VHDL, Verilog, and Mixed-HDL Simulation
for PCs Running Windows 95 & Windows NT

Model*Sim* /VHDL, Model*Sim* /PLUS, and Model*Sim* /VLOG are produced by Model Technology Incorporated. Unauthorized copying, duplication, or other reproduction is prohibited without the written consent of Model Technology.

The information in this manual is subject to change without notice and does not represent a commitment on the part of Model Technology. The program described in this manual is furnished under a license agreement and may not be used or copied except in accordance with the terms of the agreement. The online documentation provided with this product may be printed by the end-user. The number or copies that may be printed is limited to the number of licenses purchased.

Model*Sim* is a trademark of Model Technology Incorporated. PostScript is a registered trademark of Adobe Systems Incorporated. UNIX is a registered trademark of AT&T in the USA and other countries. FLEXlm is a trademark of Globetrotter Software, Inc. IBM, AT, and PC are registered trademarks, AIX and RISC System/6000 are trademarks of International Business Machines Corporation. Windows is a trademark, Microsoft and MS-DOS are registered trademarks of Microsoft Corporation. OSF/Motif is a trademark of the Open Software Foundation, Inc. in the USA and other countries. SPARC is a registered trademark and SPARCstation is a trademark of SPARC International, Inc. Sun Microsystems is a registered trademark, and Sun, SunOS and OpenWindows are trademarks of Sun Microsystems, Inc. All other trademarks and registered trademarks are the properties of their respective holders.

Copyright (c) 1990-1998, Model Technology Incorporated.
All rights reserved. Confidential. Online documentation may be printed by licensed customers of Model Technology Incorporated for internal business purposes only.

Software Version: 4.7
Published: April 1998

Model Technology Incorporated
8905 SW Nimbus Avenue, Suite 150
Beaverton OR 97008-7100 USA

phone: 503-641-1340
fax: 503-526-5410
email: support@model.com, sales@model.com
home page: http://www.model.com

$50 US

Software License Agreement

This is a legal agreement between you, the end user, and Model Technology Incorporated (MTI). By opening the sealed package, or by signing this form, you are agreeing to be bound by the terms of this agreement. If you do not agree to the terms of this agreement, promptly return the unopened package and all accompanying items to the place you obtained them for a full refund.

Model Technology Software License

1. LICENSE. MTI grants to you the **nontransferable, nonexclusive** right to use one copy of the enclosed software program (the "SOFTWARE") for each license or hardware security key you have purchased. The SOFTWARE must be used on the computer hardware server equipment that you identified in writing by make, model, and workstation or host identification number and the equipment served, in machine-readable form only , as allowed by the authorization code provided to you by MTI or its agents. All authorized systems must be used within the country for which the systems were sold. Workstation licenses must be located at a single site, i.e. within a one-kilometer radius (and identified in writing to MTI). The restriction to identified products does not apply to PC products licensed by a hardware security key, and such PC products may be relocated within the country for which sold.

2. COPYRIGHT. The SOFTWARE is owned by MTI (or its licensors) and is protected by United States copyright laws and international treaty provisions. Therefore you must treat the SOFTWARE like any other copyrighted material, except that you may either (a) make one copy of the SOFTWARE solely for backup or archival purposes, or (b) transfer the SOFTWARE to a single hard disk provided you keep the original solely for backup or archival purposes. You may not copy the written materials accompanying the SOFTWARE.

3. USE OF SOFTWARE. The SOFTWARE is licensed to you for internal use only. You shall not conduct benchmarks or other evaluations of the SOFTWARE without the advance written consent of an authorized representative of MTI. You shall not sub-license, assign or otherwise transfer the license granted or the rights under it without the prior written consent of MTI or its applicable licensor. You shall keep the SOFTWARE in a restricted and secured area and shall grant access only to authorized persons. You shall not make software available in any form to any person other than your employees whose job performance requires access and who are specified in writing to MTI. MTI may enter your business premises during normal business hours to inspect the SOFTWARE, subject to your normal security.

4. PERMISSION TO COPY LICENSED SOFTWARE. You may copy the SOFTWARE only as reasonably necessary to support an authorized use. Except as permitted by Section 2, you may not make copies, in whole or in part, of the SOFTWARE or other material provided by MTI without the prior written consent of MTI. For such permitted copies, you will include all notices and legends embedded in the SOFTWARE and affixed to its medium and container as received from MTI. All copies of the SOFTWARE, whether provided by MTI or made by you, shall remain the property of MTI or its licensors.

You will maintain a record of the number and location of all copies of the SOFTWARE made, including copes that have been merged with other software, and will make those records available to MTI or its applicable licensor upon request.

5. TRADE SECRET. The source code of the SOFTWARE is trade secret or confidential information of MTI or its licensors. You shall take appropriate action to protect the confidentiality of the SOFTWARE and to ensure that any user permitted access to the SOFTWARE does not provide it to others. You shall take appropriate action to protect the confidentiality of the source code of the SOFTWARE. You shall not reverse-assemble, reverse-compile or otherwise reverse-engineer the SOFTWARE in whole or in part. The provisions of this section shall survive the termination of this Agreement.

6. TITLE. Title to the SOFTWARE licensed to you or copies thereof are retained by MTI or third parties from whom MTI has obtained a licensing right.

7. OTHER RESTRICTIONS. You may not rent or lease the SOFTWARE. You shall not mortgage, pledge or encumber the SOFTWARE in any way. You shall ensure that all support service is performed by MTI or its designated agents. You shall notify MTI of any loss of the SOFTWARE.

8. TERMINATION. MTI may terminate this Agreement, or any license granted under it, in the event of breach or default by you. In the event of such termination, all applicable SOFTWARE shall be returned to MTI or destroyed.

9. EXPORT. You agree not to allow the MTI SOFTWARE to be sent or used in any other country except in compliance with this license and applicable U.S. laws and regulations. If you need advice on export laws and regulations, you should contact the U.S. Department of Commerce, Export Division, Washington, DC 20230, USA for clarification.

Important Notice

Any provision of Model Technology Incorporated SOFTWARE to the U.S. Government is with "Restricted Rights" as follows: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraphs (a) through (d) of the Commercial Computer-Restricted Rights clause at FAR 2.227-19 when applicable, or in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clauses in the NASA FAR Supplement. Any provision of Model Technology documentation to the U.S. Government is with Limited Rights. Contractor/ manufacturer is Model Technology Incorporated, Suite 150, 8905 SW Nimbus Avenue, Beaverton, Oregon 97008 USA.

Limited Warranty

LIMITED WARRANTY. MTI warrants that the SOFTWARE will perform substantially in accordance with the accompanying written materials for a period of 30 days from the date of receipt. Any implied warranties on the SOFTWARE are limited to 30 days. Some states do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you.

CUSTOMER REMEDIES. MTI's entire liability and your exclusive remedy shall be, at MTI's option, either (a) return of the price paid or (b) repair or replacement of the SOFTWARE that does not meet MTI's Limited Warranty and which is returned to MTI. This Limited Warranty is void if failure of the SOFTWARE has resulted from accident, abuse or misapplication. Any replacement SOFTWARE will be warranted for the remainder of the original warranty period or 30 days, whichever is longer.

NO OTHER WARRANTIES. MTI disclaims all other warranties, either express or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose, with respect to the SOFTWARE and the accompanying written materials. This limited warranty gives you specific legal rights. You may have others, which vary from state to state.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES. In no event shall MTI or its suppliers be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use of or inability to use these MTI products, even if MTI has been advised of the possibility of such damages. Because some states do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

Table of Contents

Software License Agreement . . . 3
Model Technology Software License . . . 3
Important Notice . . . 5
Limited Warranty . . . 5

1 - Introduction (p11)

Software versions . . . 11
Standards supported . . . 12
Assumptions . . . 13
Sections in this guide . . . 13
Text conventions . . . 14
HDL and HDL item defined . . . 14
Syntax conventions . . . 14
Where to find our documentation . . . 15
Download a free PDF reader . . . 15
Comments . . . 16

2 - Model*Sim* PE Installation (p17)

Current customers information . . . 18
System requirements for Model***Sim*** PE . . . 18
Installation procedure . . . 19
Install Model***Sim*** on Your Hard Drive - From CD-ROM . . . 19
Install the Security Key . . . 19
Authorization Codes . . . 19
Authorization Codes: Model Technology direct sales . . . 19
Authorization Codes: Model Technology OEM sales . . . 19
Authorization Codes: updates . . . 20
Authorization Codes: entering your permanent code . . . 20
To Run from the CD-ROM . . . 20

3 - Model*Sim* PE Graphic Interface (p21)

Model***Sim*** application window . . . 22
Model***Sim*** menu bar . . . 24
Model***Sim*** tool bar . . . 27

The Simulate a Design dialog box 28
Tree windows 29
Window overview 31
Transcript window 32
Transcript window status bar 32
Transcript window keyboard actions 33
Dataflow window 35
Tracing HDL items with the Dataflow window 36
Dataflow window mouse actions 36
List window 37
List window menu bar 38
List window status bar 39
List window mouse actions 39
Selecting HDL items to list 40
Selecting List window triggering options 40
Formatting a list 41
Editing a list 43
Saving a listing 43
Saving a List window configuration 43
Restoring a List window configuration 44
Updating the List window 44
Process window 45
Process window status bar 46
Process window mouse actions 46
Signals window 47
Signals window status bar 47
Signals window mouse actions 48
Selecting HDL item types to view 48
Forcing HDL item values 49
Adding HDL items to other windows 51
Source window 52
Source window status bar 53
Selecting the source file 53
Editing breakpoints 54
Examining values at breakpoints 54
Customizing the Source window 54
Structure window 56
Structure window status bar 58
Structure window mouse actions 58

Variables window 59
Variables window status bar 60
Variables window mouse actions 60
Wave window 61
Wave window menu bar 62
Wave window status bar 63
Wave window mouse actions 63
Adding HDL items to the Wave window 65
Editing HDL items in the Wave window 65
Formatting the Wave window 65
Analog formatting 68
Literal formatting 68
Logic Formatting 69
Zooming the Wave window 69
Using the Wave window cursors 70
Writing a PostScript file 71
Saving the Wave window configuration 72
Customizing Model*Sim* windows 72
Changing window colors 73
Changing window fonts 73
Saving your window settings 74
Keyboard shortcuts 76

4 - Tutorial: Using Model*Sim* PE (p77)

Assumptions 77
Tutorial setup 78
Basic VHDL simulation 79
Debugging a VHDL design 93
Basic Verilog simulation 105
Mixed VHDL/Verilog simulation 120
Learning more about Model*Sim*'s windows 127
Continuing with Model*Sim* PE 129

A - Help, Updates, and Licensing (p131)

Help 131

Updates 132
ModelSim PE Licensing 132
Getting help 134

B - Resources (p135)

Books 135
Organizations 137
Corporations & Consultants 138
Online resources 141

Index (p143)

1 - Introduction

Chapter contents

Software versions 11

Standards supported 12

Assumptions 13

Sections in this guide 13

HDL and HDL item defined 14

Where to find our documentation 15

The purpose of this guide is to help you install and become familiar with Model*Sim* PE, Model Technology's HDL-simulation software for PCs. Tutorials are included that apply to one or more of the following Model*Sim* simulators:

- **Model*Sim* PE/PLUS**
  VHDL, Verilog, and mixed VDHL/Verilog simulation for PCs
- **Model*Sim* PE/VHDL**
  VHDL simulation for PCs
- **Model*Sim* PE/VLOG**
  Verilog simulation for PCs

Software versions

This documentation was written to support Model*Sim* PE 4.7 for PCs. If the software you are using is a later release, check the README file on the CD-ROM that accompanied the software. Any supplemental information will be there.

Although this guide covers both VHDL and Verilog simulation, you will find it a useful reference even if your design work is limited to a single HDL.

Standards supported

Model*Sim* VHDL supports both the IEEE 1076-1987 and 1076-1993 VHDL standards. Any design developed with Model*Sim* will be compatible with any other VHDL system that is compliant with either IEEE Standard 1076-1987 or 1076-1993.

Model*Sim* Verilog is based on the IEEE Std 1364-1995 (*IEEE Standard Hardware Description Language Based on the Verilog Hardware Description Language*). The Open Verilog International *Verilog LRM version 2.0* is also applicable to a large extent. The PLI is supported on PCs and workstations, while VCD support is available for workstation users only.

In addition, all products support SDF 1.0, 2.0, and 2.1, VITAL 2.2b, and VITAL ’95.

Assumptions

We assume that you are familiar with the use of your operating system. If you are not familiar with Microsoft Windows, we recommend that you work through the tutorials provided with MS Windows before using Model*Sim*.

In addition, we assume that you have a working knowledge of VHDL and Verilog. Although Model*Sim* is an excellent tool to use while learning HDL concepts and practices, this guide is not written to support that goal. If you need more information about HDLs, visit the Model Technology home page at http://www.model.com. You can also contact your local Model Technology distributor for information about training classes.

Sections in this guide

In addition to this introduction, you will find the following major sections in this guide:

2 - ModelSim PE Installation (p17)

3 - ModelSim PE Graphic Interface (p21)

4 - Tutorial: Using ModelSim PE (p77)

A - Help, Updates, and Licensing (p131)

B - Resources (p135)

Text conventions

Text conventions used in this manual include:

italic text	provides emphasis and sets off file and path names
bold text	indicates commands, command options, and menu choices, as well as package and library logical names
`monospaced type`	monospace type is used for program and command examples
The right angle (>)	is used to connect menu choices when traversing menus as in: **File > Save**

HDL and HDL item defined

"HDL" refers to either VHDL or Verilog when a specific language reference is not needed. Depending on the context, "HDL item" can refer to any of the following:

- **VHDL**
  block statement, component instantiation, constant, generate statement, generic, package, signal, or variable
- **Verilog**
  function, module instantiation, named fork, named begin, net, task, or register variable

Syntax conventions

The syntax elements of Model*Sim* commands are signified as follows:

< >	angled brackets surrounding a syntax item indicate a user-defined argument; do not enter the brackets in commands
[ ]	square brackets indicate an optional item; if the brackets surround several words, all must be entered as a group; the brackets are not entered
. . .	an ellipsis indicates items that may appear more than once; the ellipsis itself does not appear in commands.
|	the vertical bar indicates a choice between items on either side of it. Do not include the bar in the command
#	comments are preceded by the number sign (#)

Where to find our documentation

Model Technology's documentation is available in the following formats and locations:

Document	Format	How to get it
Getting Started (installation & tutorial)	paper	shipped with the product; additional copies at $50 each (for customers with current maintenance)
Getting Started PE	PDF online	find "getstart.pdf" in the "modelsim\docs" directory on the CD-ROM, or hard drive after installation; current version available from the Support page of our web site: http://www.model.com
PE Reference Manual	paper	shipped with product; additional copies at $50 each (for customers with current maintenance)
PE Reference Manual	PDF online	find "manual.pdf" in the "modelsim\docs" directory on the CD-ROM, or hard drive after installation; current version available for ftp from the Support page of our web site: http://www.model.com (password required)
Verilog PLI tech note	ASCII	find the "vlog_pli" technote in the "modelsim/docs" directory on the CD-ROM, or hard drive after installation

Download a free PDF reader

Model Technology's online documentation (in PDF format) requires a free Adobe Acrobat reader available through http://www.adobe.com.

Comments

Comments and questions about this manual and Model*Sim* software are welcome. Call, write, or fax or email:

Model Technology Incorporated
8905 SW Nimbus Avenue, Suite 150
Beaverton OR 97008-7100 USA

phone: 503-641-1340
fax: 503-526-5410

email: manuals@model.com
home page: http://www.model.com

2 - ModelSim PE Installation

Chapter contents

Current customers information 18

System requirements for ModelSim PE 18

Installation procedure 19
- Install ModelSim on Your Hard Drive - From CD-ROM 19
- Install the Security Key 19
- Authorization Codes 19

To Run from the CD-ROM 20

This chapter covers installation of Model*Sim* PE for PCs.

Please refer to the installation instructions packaged with your product for the latest information; *they may supercede the installation instructions here.* Also please be sure to read the *readme* file included with Model*Sim* after you finish your installation.

How to protect your investment

Keep your investment in Model*Sim* safe by protecting your hardware security key. It is an absolutely essential part of your system since the software will not run without it. And while Model Technology will replace a defective key, lost or stolen keys will not be replaced. One way you can protect your investment is by insuring it. While most insurance companies won't insure software, they will insure hardware, so insure the key for the full Model*Sim* value.

Keys and time

Model*Sim* security keys for evaluations, or short-term licenses contain a time-based code. If you set your PC's system date back or move the key to a PC with an earlier system date, the key will not work until the new date "catches up" with the original date. System dates do not effect permanent licenses.

Current customers information

If you are using V-System PC version 4.4 or older, you should install Model*Sim* 4.7 in different directory.

You will also need to regenerate your design libraries with **-refresh** before running any simulations. By default, the **work** library is updated; use **-work <library>** to update a different library. For example, if you have a library named **mylib** that contains both VHDL and Verilog design units you would use both of these commands:

```
vcom -work mylib -refresh

vlog -work mylib -refresh
```

You don't need to regenerate the std, ieee, vital22b, and verilog libraries shipped with this release.

System requirements for Model*Sim* PE

Model*Sim* PE requires:

- a 386/486/Pentium PC
- 32Mb (megabytes) of memory
- Microsoft Windows 95, or Windows NT (not supplied)
- a hard disk drive with more than 10 Mb spare capacity
- a CD-ROM drive

Installation procedure

Install Model*Sim* on Your Hard Drive - From CD-ROM

The CD-ROM will auto-start when you put it in your reader; then follow the directions that appear on your screen. Or if the CD-ROM is already in the reader, double-click on the Model*Sim* CD icon to start.

Install the Security Key

Model*Sim* version 4.7 requires a hardware security key with an authorization code. Install the key on the parallel printer port. If you have a printer or another device attached to the same port, attach that device to the security key. It will not interfere with normal operations. Please remember to protect your key; Model Technology will replace defective keys, but not lost keys.

Authorization Codes

Your key's authorization code determines whether VHDL, Verilog, or mixed-HDL support is turned on. All codes are based on the product type you purchased, the key ID, and your maintenance expiration date.

Authorization Codes: Model Technology direct sales

Your key should be delivered already programmed with your permanent code. If it is not, send email to license@model.com, with your key number and return email address.

Authorization Codes: Model Technology OEM sales

If you purchased from a Model Technology OEM partner, you received a 30-day code. Obtain your permanent code by contacting your OEM licensing department with your key number and email address. They will forward the request to Model Technology and you will receive your permanent code by email.

Authorization Codes: updates

If you receive a new programmable key to replace an old-style, non-programmable key, your new key is delivered ready to run for 30 days from when you first invoke Model*Sim.* You can get a permanent code for the product you purchased by returning your old key along with your new key's serial number and your email address. We will send you a permanent code via email.

Authorization Codes: entering your permanent code

To enter a new authorization code, make this menu selection when you start Model*Sim*: **Help > About ModelSim > Authorization**. (If your authorization has expired Model*Sim* will prompt you with a dialog box.) Key in your new code and click **Apply**; the authorization information changes to reflect the current type of authorization and the new maintenance end date. Click **Ok** to complete the authorization.

When it is permanently programmed, your key will run all products of the correct type with build dates before your maintenance end date. When your maintenance expires you can still run any product built up until that date. When you renew your maintenance, you will receive a new authorization code that will extend the date.

Updates to Model*Sim* PE will be made available via the postal system, ftp, and our website at http://www.model.com. Information about changes to the PE software will be posted on our website as they occur. We hope you find this distribution system of electronic updates and key-authorized software efficient and easy to use.

To Run from the CD-ROM

A complete installation of Model*Sim* PE is included on the CD-ROM allowing you to run Model*Sim* directly from the CD-ROM. Just double-click on *modelsim.exe*. (You will have to create your **work** library on a different disk, of course!) If you are running Windows NT, you will need to have the security key drivers installed. If you have already installed them, or are running Windows 95, you can run Model*Sim* directly from the CD-ROM, although it is recommended you install Model*Sim* on your hard drive.

3 - ModelSim PE Graphic Interface

Chapter contents

ModelSim application window 22

The Simulate a Design dialog box 28

Tree windows 29

Window overview 31
- Transcript window 32
- Dataflow window 35
- List window 37
- Process window 45
- Signals window 47
- Source window 52
- Structure window 56
- Variables window 59
- Wave window 61

Customizing ModelSim windows 72

Keyboard shortcuts 76

This chapter provides a reference to the Model*Sim* windows you will use to examine and debug your VHDL and Verilog designs.

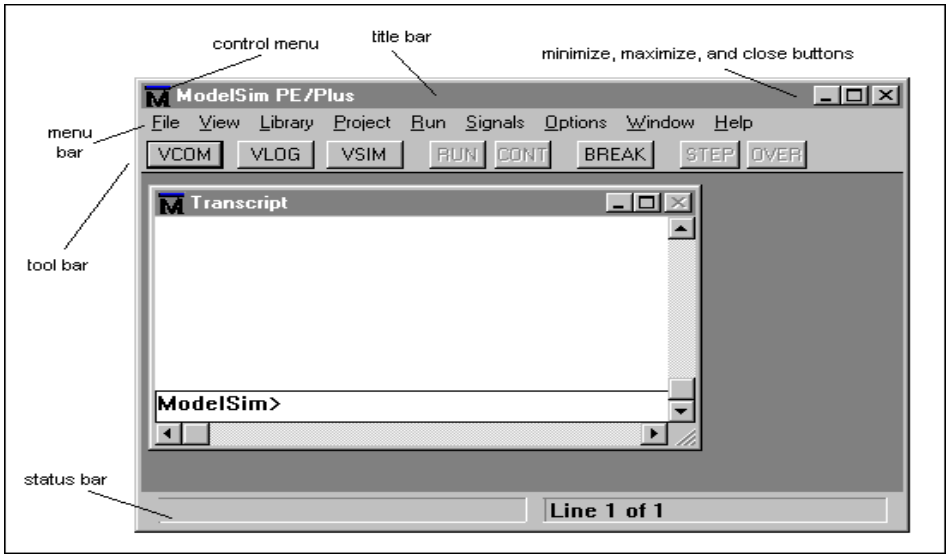
Terms

The Model*Sim* PE/PLUS environment makes use of the Microsoft Windows conventions and vocabulary for windows, menus, mouse buttons, icons, etc. If you're not familiar with these, please work through the Windows tutorial provided by Microsoft. In this chapter, the term *application window* is used as it is defined by Microsoft; it is the window in which Model*Sim* is running. The windows created by Model*Sim* within the application window are simply called *windows*.

Model*Sim* application window

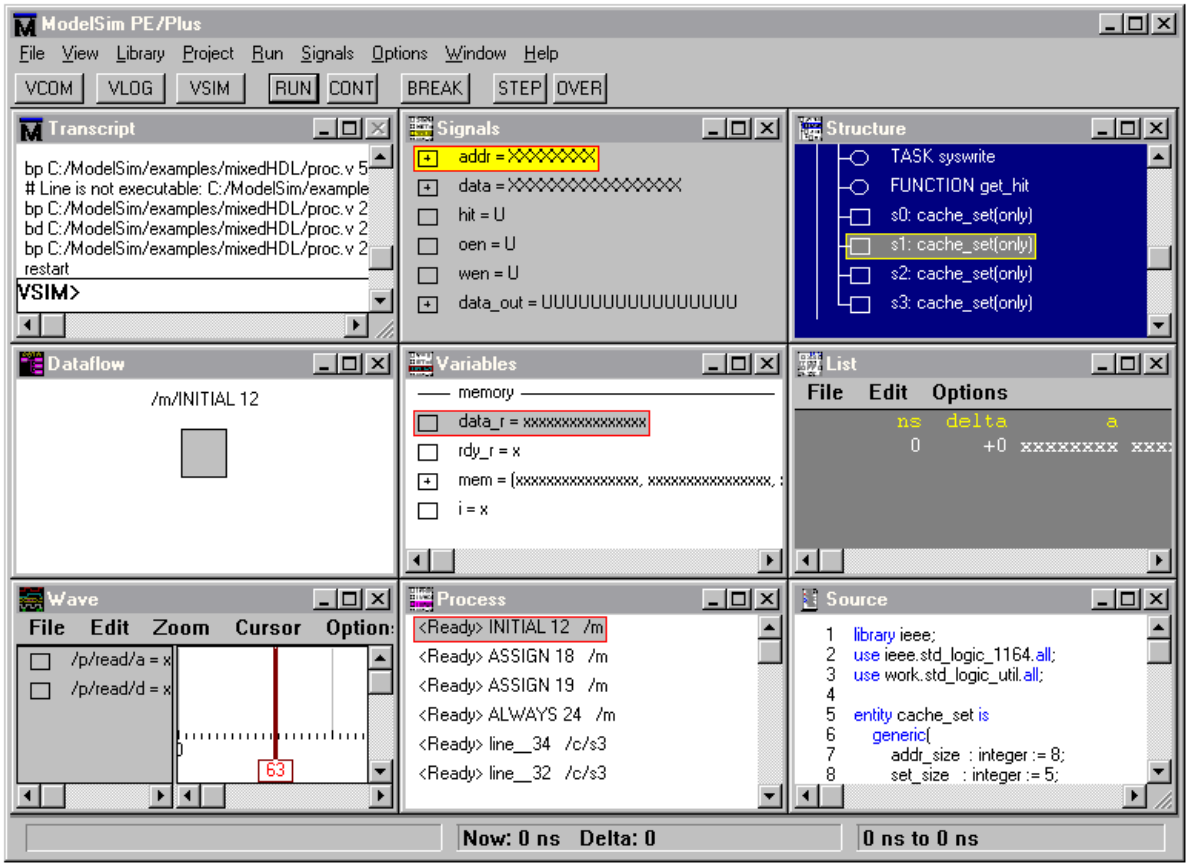
The Model*Sim* application window and Transcript window are pictured below as they appear when Model*Sim* is first started.

Note that typical window parts are used in Model*Sim*. The control-menu icon, title bar, minimize, maximize and close buttons appear across the top. A menu bar appears immediately under those items. Scroll bars are available when necessary at the right side and bottom of the window.



A tool bar (under the menu bar) and a status bar (at the bottom of the application window) are added by Model*Sim*. The status bar gives you information about the data in the active Model*Sim* window. The tool bar gives you access to most common Model*Sim* functions no matter which Model*Sim* window is active. Choices under the **Window > menu bar** option let you turn on and off the tool bar and the status bar.

In addition to the Transcript window, the Model*Sim* application window also accommodates eight other windows for use during your design compilation, simulation, and debugging. They are the Dataflow, List, Signals, Source, Structure, Process, Variables, and Wave windows. A portion of each window is shown below.



Model*Sim* menu bar

The menu bar at the top of the application window lets you access all the Model*Sim* commands and features. The menus are listed below with brief descriptions of the command's use.

File

Directory	change working directories
Execute Macro	run a macro
Compile VHDL	compile VHDL source
Compile Verilog	compile Verilog source
Simulate	start a simulation
Restart	restart a simulation
End Simulation	end a simulation and stay in Model*Sim*
Select Source File	select the code shown in Source window
Exit ModelSim	exit Model*Sim*

View

All	open all VSIM windows
<window_name>	select a Model*Sim* <window_name> to open and/or view

Library

New	create a new design library
Contents	view the contents of a library
Mapping	map a library to a new name

Project

New	create a new modelsim.ini project file
Change	change to another project file
<former project files>	shows up to three former modelsim.ini files
<current project file>	name of current modelsim.ini file is checked

Run

Run <default run length>	run simulation for one default run length
Run Forever	run simulation until you stop it
Step	single-step the simulator
Step Over	execute without single-stepping through a subprogram
Cont	continue the simulation
Break	stop the simulation

Signals

View	select which HDL item types to view
Add to Wave	add Selected HDL items, items in Region, items in Design, or items from the Dataflow window to the Wave window
Add to List	add Selected HDL items, items in Region, items in Design , or items from the Dataflow window to the List window
Add to log file	add Selected HDL items, items in Region, or items in Design to a log file
Force Signal	change value of selected HDL item

Options

Compile Options	change compiler options; select either VHDL (vcom) or Verilog (vlog) compiler
Simulation Options	select simulation and signal display options

Edit Breakpoints	modify or delete source code breakpoints
Macro Options	Echo or Hide the macro commands in the Transcript window
Processes	view Active processes, or All processes in the current region

Window

Cascade	cascade windows
Tile Horizontal	tile windows horizontally
Tile Vertical	tile windows vertically
Arrange Icons	arrange icons at bottom of application window
Restore All	restore all icons to windows
Colors	change colors of window features
Fonts	change font settings in windows
Settings	store window settings
Tool Bar	toggle tool bar
Status Bar	toggle status bar
<window_name>	switch to, or show the selected Model*Sim* <window_name> if hidden

Help

Index	switches to the Model*Sim* help index
Using Help	switches to the Windows help introduction
About ModelSim	display Model*Sim* application information

Model*Sim* tool bar

In addition to the functions accessed through the menu bar, you can use the Model*Sim* tool bar for several frequently-used commands::

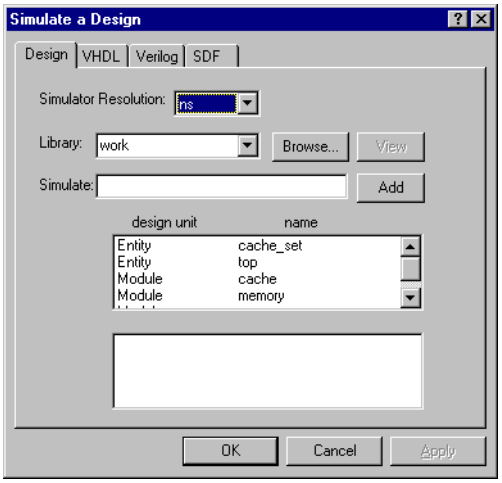
Button	Menu equivalent	VSIM command equivalent
VCOM	File > Compile VHDL	vcom
VLOG	File > Compile Verilog	vlog
VSIM	File > Simulate	vsim
RUN	Run > Run <default_run_length>…	run
CONT	Run > Cont….	continue
BREAK	none	*
STEP	Run > Step….	step
OVER	Run > Step Over….	step -over

Tool bar buttons are grayed out (like **RUN** above) when they are not available.

The Model*Sim* menu bar commands are accessible with the Windows keyboard command equivalents (<Alt> commands).

The Simulate a Design dialog box

When you first invoke VSIM you will see the **Simulate a Design** dialog box below.

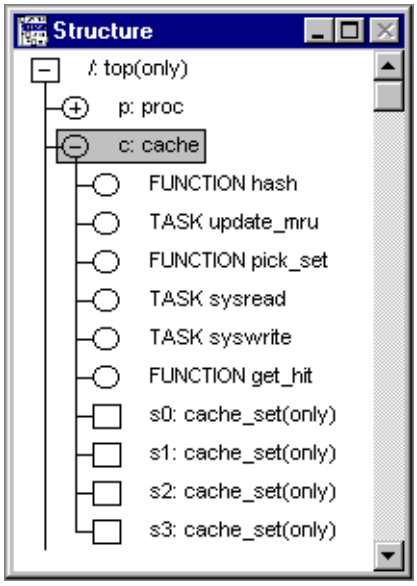


This dialog box has four tabs to select simulation options:

- The first, labeled **Design**, allows you to select a library and top-level design unit to simulate.
- The second tab, labeled **VHDL,** allows setting of several VHDL-specific options.
- The **Verilog** tab supports Verilog-specific options including control of time and hazard checking as well as user-defined simulator arguments.
- The fourth tab configures **SDF** (Standard Delay Format) functions.

Tree windows

In many windows, Model*Sim* provides a hierarchical or “tree view” of your design. For example, in the Structure window, one level of the hierarchy is created by each VHDL and Verilog HDL item. This type of view is used in the Structure (shown), Signals, Variables, and Wave windows.



HDL items

The following HDL items for VHDL and Verilog are represented by hierarchy within tree windows:

- **VHDL**
  component instantiation, generate statement, block statement, and package
- **Verilog**
  module instantiation, named fork, named begin, task and function

When you see this view, you can use the mouse to collapse and expand levels by clicking on the plus and minus boxes. Note - within the Structure window only - VHDL items are indicated with a box and Verilog items are indicated with a circle.

[+] A plus box or circle before a name indicates that you can expand the item to view the lower level elements it contains. You do this by clicking the box containing the plus sign.

[-] A minus box or circle before a name indicates that the item has been expanded and that you can hide the names of the lower level elements it contains. You do this by clicking on the box containing the minus sign.

[ ] An empty box or circle indicates that the item contains no lower-level elements.

Clicking on an item name - as opposed to the box or circle - will select that item. Clicking on a second item will select the new item and deselect the first item. In the Wave and Signals windows you can make multiple selections of HDL item names; control-click to add selections, shift-click to extend selections.

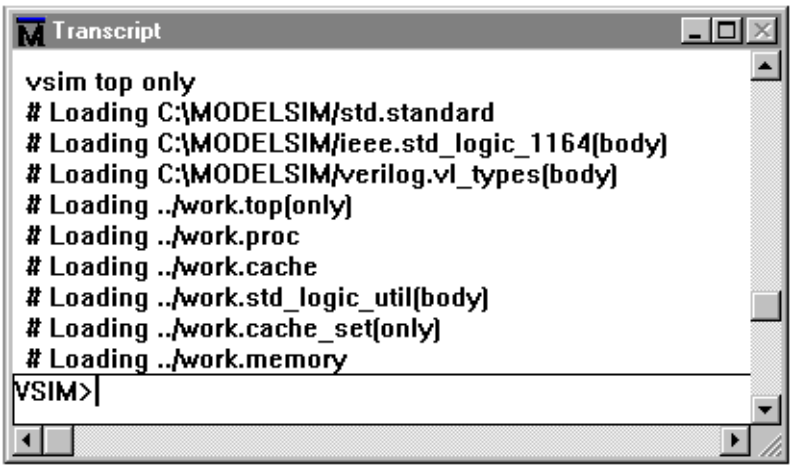
Window overview

Once you invoke VSIM nine windows become available for use during simulation. These windows are:

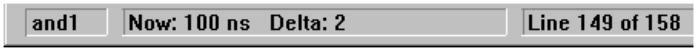
- Transcript window (p32)
  The command-line window; displays a transcript of all command activity.
- Dataflow window (p35)
  Lets you trace signals and nets through your design by showing related processes.
- List window (p37)
  Shows the simulation values of selected VHDL signals, and Verilog nets and register variables in tabular format.
- Process window (p45)
  Displays a list of processes that are scheduled to run during the current simulation cycle.
- Signals window (p47)
  Shows the names and current values of VHDL signals, and Verilog nets and register variables in the region currently selected in the Structure window.
- Source window (p52)
  Displays the HDL source code for the design.
- Structure window (p56)
  Displays the hierarchy of structural elements such as VHDL component instances, packages, blocks, generate statements, and Verilog model instances, named blocks, tasks and functions.
- Variables window (p59)
  Displays VHDL constants, generics, variables, and Verilog register variables in the current process and their current values.
- Wave window (p61)
  Displays waveforms, and current values for the VHDL signals, and Verilog nets and register variables you have selected.

Transcript window

The Transcript window is the first window to appear when you use Model*Sim*. If you tend to use the menus to control Model*Sim*, you can almost ignore the Transcript window; just think of it as the place where your menu commands are recorded. If on the other hand, you are command-line oriented, then you can use the Transcript window to control most Model*Sim* functions. Even for GUI types, sometimes it's simpler to type a **run** than to reach for the mouse and hunt down the equivalent menu bar command.



Transcript window status bar



The Transcript window status bar has three fields that display different information depending on the current Model*Sim* task. When the application is first started, the status bar shows:

- the application name "Model*Sim* PE/ (PLUS, VLOG or VHDL)" in the first field (on the left)
- nothing in the second field

- the number of the top line visible in the Transcript window and the total number of lines in the right most field

When a circuit is compiled, the status bar shows:

- the filename being compiled
- the line number of the current line being compiled (updated every 100 lines)
- the Transcript window line information

The status bar shown above indicates that a circuit is loaded for simulation; it reads from left to right:

- the current region (**and1** in this case),
- the current simulation time and delta (100 ns and 2),
- the numbers of the top line visible in the Transcript window and the total lines; this field works in conjunction with the transcript window vertical scroll bar (line 149 is at top of the window and there are a total of 158 lines in the transcript).

Transcript window keyboard actions

Since the Transcript window is the main Model*Sim* control window, typing at any time - even while another window is active - is the equivalent of typing on the Transcript window command line.

Transcript window command line shortcuts

On the Transcript window command line, you can also use these shortcuts:

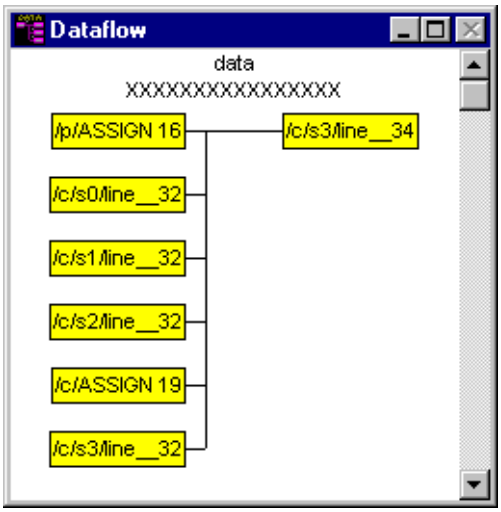
Key	Action
<Home>	Move to beginning of the line.
<End>	Move to the end of the line.
<up arrow>	Retrieves the previous command. You can press this key repeatedly to back up through the command history.
<down arrow>	Retrieves the next command. This is used if you go back through the command history and went too far; you can go forward to the next command you typed in earlier.
<Esc>	Clears any existing characters from the command line.
<Ctrl><Insert>	Copies text from a selected area of the command line.
<Shift><Insert>	Pastes text into the command line. The text must have been previously cut using <Ctrl><Insert>.
<Shift><Delete>	Cut text.
<F8>	Completes a line when you have typed a few characters. Model*Sim* searches for the most recent command that matches the characters. Using <F8> again retrieves earlier matches.

In addition, the command line is fully editable; you can move the cursor anywhere on the line and delete or insert characters as you wish. You do not have to move the cursor to the end of the command line before pressing <Enter>; the entire command line will be submitted regardless of where the cursor is located.

Also see "Keyboard shortcuts" (p76).

Dataflow window

The Dataflow window provides a graphic view of HDL items flowing in and out of processes during your simulation run, which allows you to trace HDL items through your design. The window is opened when you use the **Window > Dataflow** menu choices or issue the **view dataflow** command



For VHDL, The Dataflow window shows either:

- a signal in the center of the window with all the processes that drive the signal on the left, and all the processes that read the signal on the right, or
- a process with all the signals read by the process show as inputs on the left of the window, and all the signals driven by the process on the right.

For Verilog, The Dataflow window shows either:

- a net in the center of the window with all the processes that drive the net on the left, and all the processes triggered by the net on the right, or
- a process with all the nets that trigger the process show as inputs on the left of the window, and all the nets driven by the process on the right.

To put a particular process in the Dataflow window, just click on that process in the Process window. The Dataflow window will also show the current process when single-stepping or when VSIM hits a breakpoint. To put a particular signal or net in the window, just click on the item in the Signal window.

Tracing HDL items with the Dataflow window

The Dataflow window is linked with the Signals and Process windows. To examine a particular process in the Dataflow window, click on the process name in the Process window.To examine a particular HDL item in the Dataflow window, click on the item name in the Signal window.

Once you have an item in center of the Dataflow window, you can:

- click once on a process name to make the Source and Variable windows update to show that process,
- click twice on a process name to move the process to the center of the Dataflow window

When you have a process in center of the Dataflow window, you can:

- click once on an item name to make the Source and Signals windows update to show that item,
- click twice on an item name to move that item to the center of the Dataflow window.

The Dataflow window will display the current process when you single-step or when VSIM hits a breakpoint.

Dataflow window mouse actions

In the Dataflow window, you can use the left mouse button to:

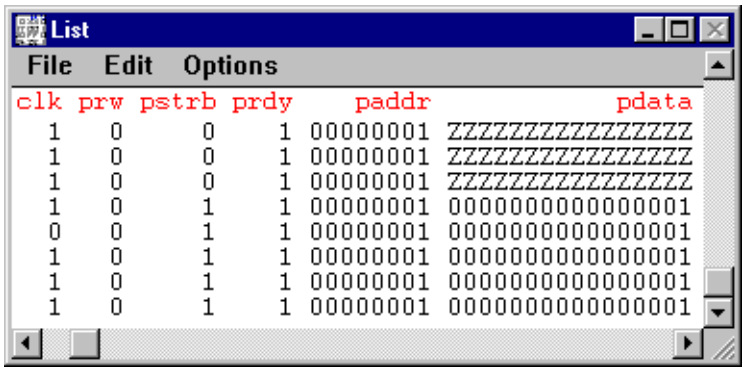
- click once on a process, to show the state of that process in the Source and Variable windows
- double-click on a process to move it to the center of the window
- double-click on a signal or net to move it to the center of the window.

List window

Two windows in Model*Sim* provide views of your simulation run. The List window shows you the results in a table format.

Commands on the List window menu bar let you:

- decide which HDL items are listed
- set the standard item properties by setting defaults
- move or delete items already in the List window
- format an item's radix, label, width, and triggering properties
- save your listing to an ASCII file
- save your List window configuration for future use



HDL items

The following HDL items for VHDL and Verilog are viewable within the List window:

- **VHDL** signals
- **Verilog** nets and register variables

List window menu bar

The following commands are available from the List window menu bar.

File

Save Listing	save the current listing to a file
Save Configuration	save the format of the List window for another time

Edit

Cut	cut selected HDL item and place it on the clipboard
Copy	copy the selected item to the clipboard
Paste	paste a cut or copied item
Before	paste item before current position
After	paste item after current position
At Beginning	paste item at the beginning of the list
At End	paste item at the end of the list
Delete All	delete all items from list window

Options

Triggering Options	specify when an HDL item triggers a line in the list
Default Options	specify default radix, label length, and triggering for all items in a list
Signal Options	specify individual item radix, label, width, and triggering
Display Options	update the List window During or After a simulation run

List window status bar

The List window status bar is illustrated below.

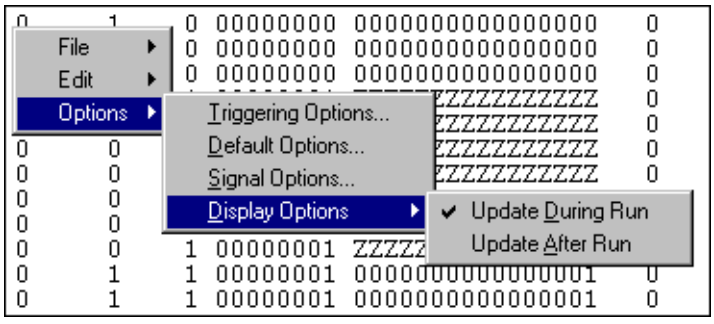
Only two fields are used in the List window status bar:

- the current simulation time and delta (300 ns and 3 in the this case)
- the line number of the list line visible at the top of the List window and the total number of lines in the list (line 4 of 12 in the illustration)

List window mouse actions

Use the mouse to select HDL items for cutting, pasting, or editing by clicking on the item names at the top of the list.

You can also use the right mouse button to call up the List window menu bar:



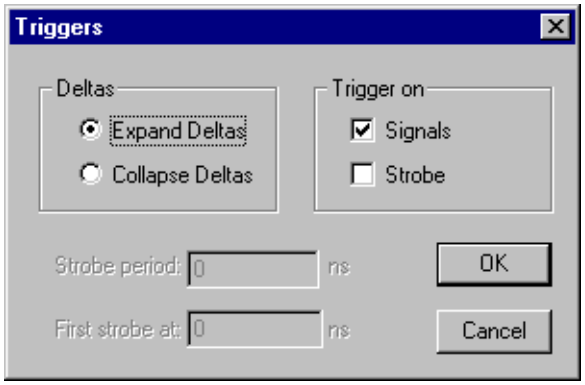
Selecting HDL items to list

You can select the items to list with the menu bar: **Signals > Add to List** as described in "Adding HDL items to other windows" (p51).

You can also type **list *** to include every HDL item in the current region specified by what's selected in the Structure window. And you can use the **list <signal_name>** command.

Selecting List window triggering options

Use the List window's menu bar command **Options > Triggering Options** to open the **Triggers** dialog box.



- **Deltas**
  If you select the **Expand Deltas** option button, a new line in the list is written for each time step on which HDL items change, including deltas within a single time step.

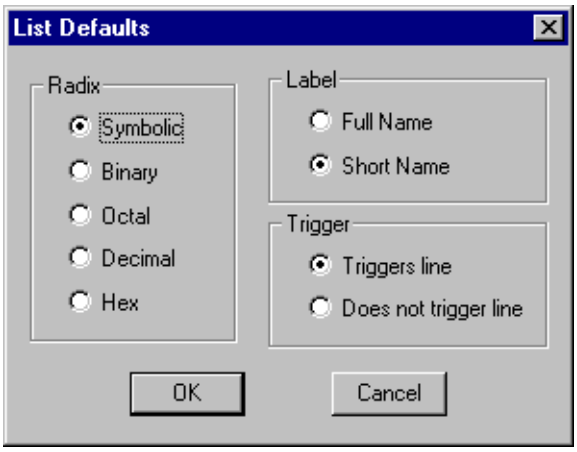
  If you select **Collapse Deltas**, only the final value for each time step is written.
- **Trigger on**
  If you choose the **Signals** check box, then a change in any item you have selected to list that has been defined as "triggers line" will cause a new line to be written in the listing.

  Choose **Strobe** if you want to list all selected items at specified time intervals. This selection enables the **Strobe period** and **First strobe at** text boxes.

Use the **OK** button when you've finished selecting your trigger options.

Formatting a list

Use List window's menu bar command **Options > Default Options** to bring up this **List Defaults** dialog box.



The List Defaults dialog box allows you to specify the default list attributes of HDL items that you add to the list window.

- **Radix**
  Allows you to specify the radix (base) in which the HDL item value is expressed. The default radix is symbolic, which means that for an enumerated type, the List window lists the actual values of the enumerated type of that item.

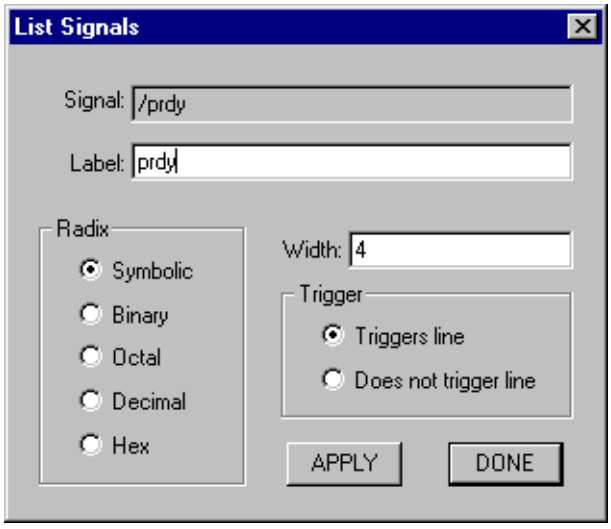
  For the other radixes—binary, octal, decimal, or hexadecimal—the item value is converted to an appropriate representation in that radix. In the system initialization file, *modelsim.ini*, you can specify the list translation rules for arrays of enumerated types for binary, octal, decimal, or hexadecimal item values in the design unit.
- **Label:**
  Allows you to specify the label that is to appear at the top of the List column for the

specified HDL item. The label may be either just the item name (**Short Name**) or the item path (**Full Name**).

- **Trigger**
  Allows you to specify whether or not a change in the value of the selected item is to trigger a new line in the listing.

You can also change the format of the HDL items after they have been added to the List window. Use the List window's menu bar command **Options > Signal Options** to bring up the **List Signals** dialog box. Note in the illustration how the item you select (by clicking on the name at the top of the column) in the List window appears in the **List Signals** dialog box so it can be reformated.

While the dialog box is open, you can select other items in the List window and change their as well.



- **Signal**
  The selected signal name.
- **Label**
  Specify the item label to use in the List window.

- **Radix**
  Change the radix for the selected signal; overrides the radix default set in the List Defaults dialog box.
- **Width**
  The specification you enter in this text box determines the number of spaces the item's column occupies in the listing.
- **Trigger**
  Specify whether or not a change in the value of the selected item is to trigger a new line in the listing ; overrides the Trigger default set in the List Defaults dialog box.

Editing a list

Once you have the HDL items you want in the list, you can order them in the listing. To determine which items appear first, second, third, etc. in your listing, use the List window menu bar **Edit** commands:

- **Cut**
- **Copy**
- **Paste > Before | After | At Beginning | At End**
- **Delete All**

Saving a listing

Use the List window's menu bar command **File > Save Listing** to save your simulation results as an ASCII file named <*filename*>.*lst*.

Be aware that this file could be very large!

Saving a List window configuration

Once you've constructed a list format that you like and you plan to use it again, or if you plan to compare two lists, it's best to make sure your format is always the same. You can do that by saving your list configuration with the List window's menu bar command: **File > Save Configuration**.

Restoring a List window configuration

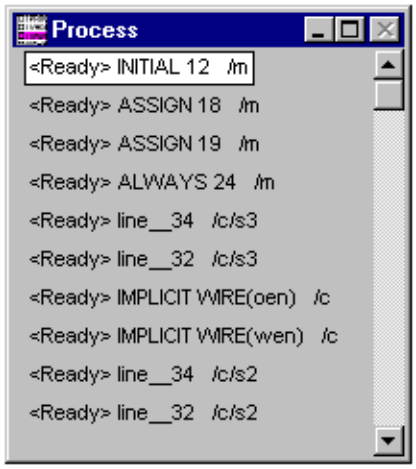
Use the Model*Sim* menu bar command **File > Execute Macro** to restore your List window configuration.

Updating the List window

You can determine when the List window is updated (either during or after simulation runs) with the List window menu bar command: **Display Options > Update During Run | Update After Run**.

Process window

The Process window displays a list of processes that are scheduled to run during the current simulation cycle and indicates the hierarchical pathname of the instance in which the process is located.



If you select **View Active Processes** with the menu bar commands, the Process window displays all the processes that are scheduled to run during the current simulation cycle. If you choose **View Processes In Region**, the window displays any processes that exist in the region that is selected in the Structure window.

Each item in the scrollbox is preceded by one of the following indicators:

<Ready>
: Indicates that the process is scheduled to be executed within the current delta time. Clicking on a Ready process schedules it to be executed next.

<Wait>
: Indicates that the process is waiting for an HDL item to change or for a specified timeout period.

<Done>
: Indicates that the process has executed a wait statement without a timeout or a sensitivity list. The process will not restart during the current simulation run.

You can debug your design (or explore someone else's) by using the Process window. When you click on a process in the Process window:

- the Structure window is updated to show the region in which the process is located
- the Variable window is updated to show the VHDL variables and Verilog register variables in the process
- the Source window changes to show the associated source code
- the Dataflow window changes to show the process, the signals and nets the process reads, and the signals and nets driven by the process.

Process window status bar

The Process window status bar is illustrated below.

Only two fields are used in the Process window status bar:

- Process window status (either Active Processes or Processes in Region) and name of the design entity
- simulation time and delta

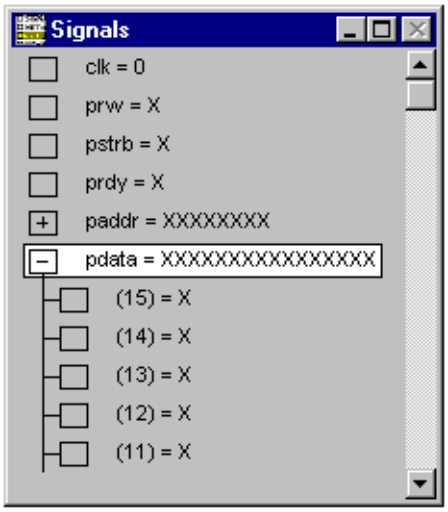
Process window mouse actions

In the Process window, you can click on a process name to:

- select the process you wish to run next
- update the Variables and Source windows
- place the selected process in the Dataflow window

Signals window

The Signals window shows the names and values of HDL items in the current region (selected in the Structure window). Items are shown in the order they are declared in the HDL source code.



The names of any composite types (arrays and record types) are shown in a hierarchical fashion, and you can expand and contract the display to view the structure of these elements.

Signals window status bar

The Signals window status bar displays two fields:

- the pathname of the current region
- the current simulation time and delta

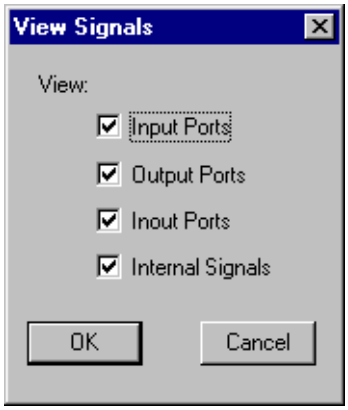
Signals window mouse actions

In the Signals window, you can use the mouse to:

- select an HDL item by clicking on its name
- <Control> <left mouse button> adds/subtracts individual selections
- <Shift> <left mouse button> extends the selections
- collapse and expand levels by clicking on the plus and minus boxes
- move a selected HDL item to the Dataflow window

Selecting HDL item types to view

Use the **Signals > View** menu bar sequence and the **View Signals** dialog box to select the HDL items you want to view in the Signals window.



You can select any combination of the following:

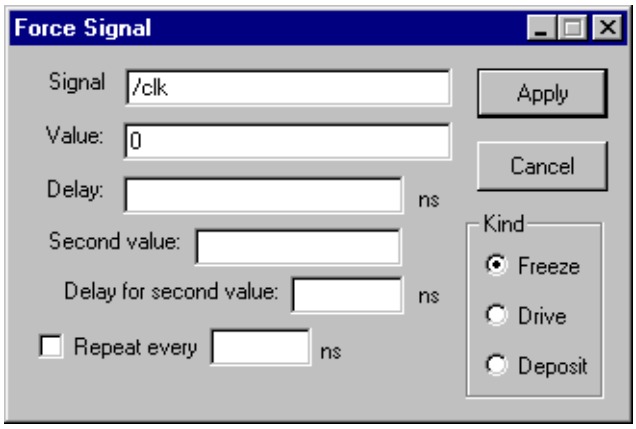
- input ports
- output ports
- inout ports
- internal items

The names of any VHDL composite types (arrays and record types) are shown in a hierarchical fashion. Hierarchy also applies to Verilog nets and vector memories. (Verilog vector registers do not have hierarchy because they are not internally represented as arrays.) To indicate hierarchy, each name is preceded by a box that contains a plus sign (+), a minus sign (-), or may be empty.

You can expand and contract the display to view the items by clicking on the boxes at the left of the Window. You can make multiple selections in the Signals window; control-click to add selections, shift-click to extend selections.

Forcing HDL item values

If you use the **Signals > Force Signal** menu bar command, you can change HDL item values. The command brings up this dialog box.



Model*Sim* offers many options beyond changing an HDL item value. With the dialog box you can also delay when the stimulus is to be applied, specify a second value to be applied, delay that value's application, and have Model*Sim* repeat your changes at the intervals you specify.

The **Force Signal** dialog box has these options:

- **Signal**
  Initially displays the selected signal or net; or you can specify another signal or net to which you want to apply a stimulus.

- **Value**:
  Initially displays the current value, which can be changed by entering a new value into the field. A value can be specified in radixes other than decimal by using the form (for VHDL and Verilog, respectively):

  ```
  base#value  -or-  b|o|d|h'value
  ```

For example, 16#EE or h'EE specifies the hexadecimal value EE.

- **Delay**:
  Allows you to specify how many time units from the current time the stimulus is to be applied.
- **Second value**:
  Allows you to specify a second value so that you can toggle a signal or net between two specifications.
- **Delay for second value**:
  Specifies the delay for the second value.
- **Repeat every**
  Allows you to specify the time interval after which the stimulus is to be repeated. A value of 0 indicates that the stimulus is not to be repeated. If you used two values, both are repeated.

Kind

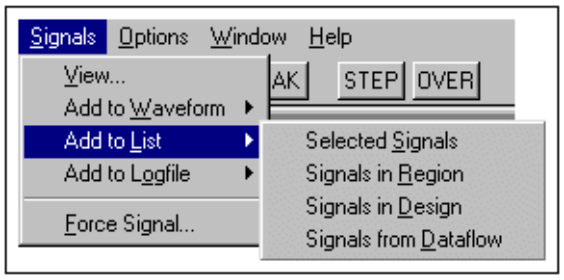
- **Freeze**
  Freezes the signal or net at the specified value until it is forced again or until it is unforced with a **noforce** command.
- **Drive**
  Attaches a driver to the HDL item and drives the specified value until the signal or net is forced again or until it is unforced with a **noforce** command. This option is illegal for unresolved VHDL signals.
- **Deposit**
  Sets the signal or net to the specified value. The value remains until there is a subsequent driver transaction, or until the signal or net is forced again, or until it is unforced with a **noforce** command.

Freeze is the default for Verilog nets and unresolved VHDL signals and **Drive** is the default for resolved signals.

If you prefer **Freeze** as the default for resolved and unresolved signals, you can change the default force kind in the *modelsim.ini* file.

Adding HDL items to other windows

From the menu bar you can use the **Signals > Add to Waveform | List | Logfile** commands to instruct Model*Sim* to add VHDL signals or Verilog nets to the List or Wave windows or to the log file (an archive file in binary format). There are menu commands for each feature.

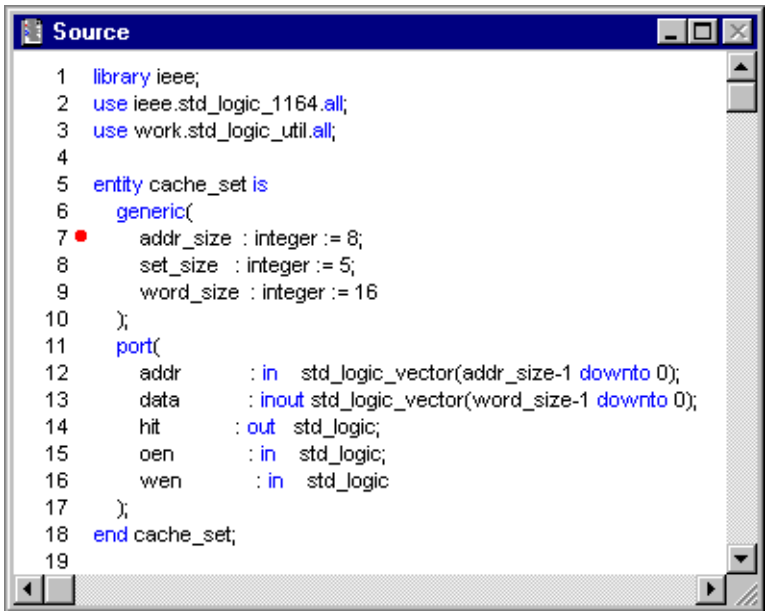


There are four choices for signal selection:

- **Selected Signal**
  Lists/logs/displays only the signal or net selected in the Signals window.
- **Signals in Region**
  Lists/logs/displays all signals and nets in the HDL item selected as the current region in the Structure window.
- **Signals in Design**
  Lists/logs/displays all signals or nets in the design.
- **Signals from Dataflow**
  Lists/logs/displays only the signal or net selected in the Dataflow window.

Source window

The Source window allows you to view your VHDL or Verilog source code in read-only mode. A marker at the left side of the window pane points to the next line to be executed.



If any breakpoints have been set, they are signified by a colored dot next to a line number. To set a breakpoint, double-click at or near the right side of the line number. The breakpoints toggle, so you can double-click again to delete an existing breakpoint. There is no limit to the number of breakpoints you can set.

Source window status bar

The Source window status bar is illustrated below.

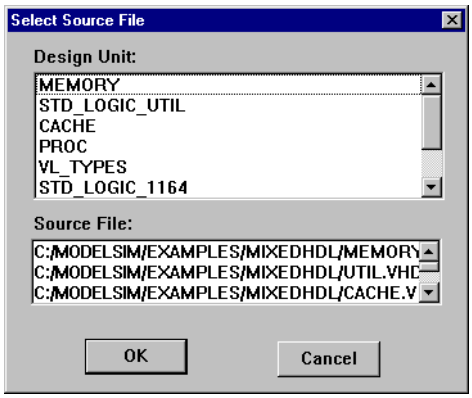
The status bar for the Source window shows (from left to right):

- the name of the current source file,
- the current simulation time and delta (100 ns and 2 in this case),
- the current line number of the source code line displayed at the top of the Source window and the total number of lines in the source code.

Selecting the source file

The Source window default file is the design you selected to simulate. An easy way to look at a file that is not currently being displayed is to use the Structure window to select a different design unit as the current working environment

You can also change the source file graphically with the **File > Select Source File** menu sequence, which brings up the **Select Source File** dialog box shown below:

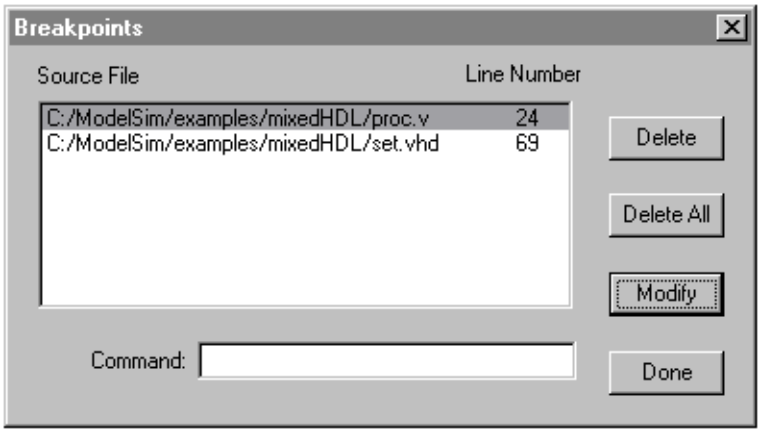


See Structure window (p56) for more information on selecting HDL items within your design.

Editing breakpoints

You can double-click anywhere in a line to set or remove a breakpoint.

Besides using the mouse to set and remove breakpoints, there is also the **Options > Edit Breakpoints** menu command. It brings up **Breakpoints** dialog box.



With the **Breakpoints** dialog box you can delete or modify any breakpoint in the VHDL or Verilog source code.

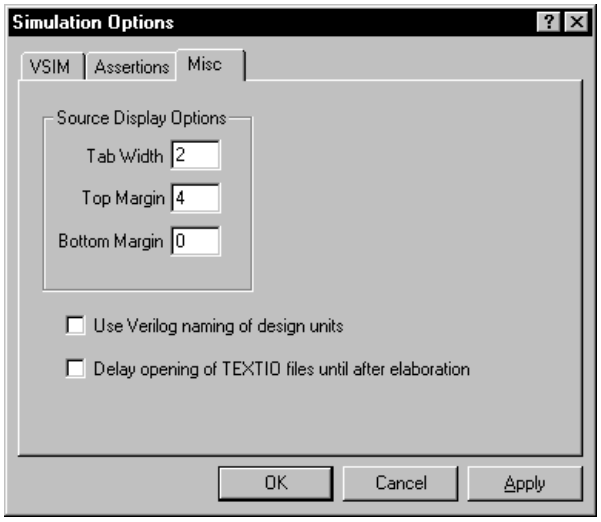
Examining values at breakpoints

Use the **Command** text box in the Breakpoints dialog box to enter a command that Model*Sim* will execute when the breakpoint is reached. A typical use is to examine a variable each time the simulation gets to a breakpoint; in a case like this you can enter a command like the example in the text box:

```
EXAMINE <variable_name>
```

Customizing the Source window

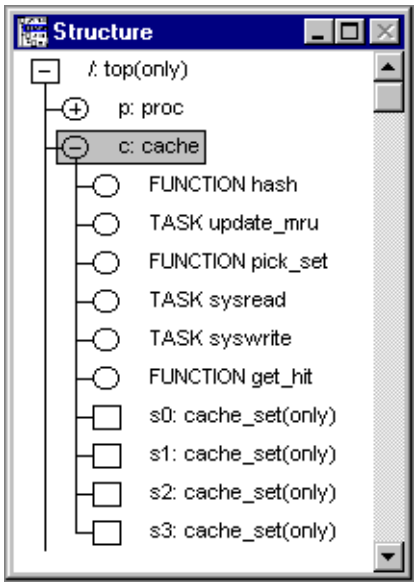
Selecting **Options > Simulation Options** brings up the Simulation Options dialog box. Select the **Misc** tab to set tabs and margins for the VHDL or Verilog code displayed in the Source Window.



This dialog box sets the context of the information shown in the Source window. When Model*Sim* scrolls to a line of source code, **Top Margin** sets the minimum number of lines displayed above the active line. **Bottom Margin** sets the number of lines shown below the active line when you single-step the simulator.

Structure window

The Structure window provides a hierarchical view of the structure of your design. A level of hierarchy is created by each HDL item within the design.



HDL items

The following HDL items for VHDL and Verilog are represented by hierarchy within Structure window:

- **VHDL**
  component instantiation, generate statement, block statement, and package
- **Verilog**
  module instantiation, named fork, named begin, task and function

Within the Structure window, VHDL items are indicated by a box and Verilog items are indicated by a circle.You can expand and contract the display to view the elements by clicking on the boxes or circles at the left of the Window.

The first line of the VSIM Structure window indicates the top-level design unit being simulated.

An instance name displayed in the Structure window consists of the following parts:

```
instantiation_label: entity(architecture)
```

where:

`instantiation_label`
Indicates the label assigned to the component instance in the component instantiation statement. The top-level component uses "/" instead of an instantiation label.

`entity`
Indicates the name of the entity that has been instantiated.

`architecture`
Indicates the name of the architecture associated with the entity.

When you select a design level (by clicking on the name), Model*Sim* updates the Source and Signals windows to match the region you selected. This feature provides a useful method for finding the source code for a selected region; the system keeps track of the pathname where the source is located and displays it automatically without the need for you to provide the pathname.The Process window will also be updated if you have it set to **View Processes in Region**.

For more information see Source window (p52), List window (p37), and Process window (p45).

Structure window status bar

The Structure window status bar is illustrated below.

Only two fields are used in the Structure window status bar:

- the pathname of the selected design unit,
- the simulation time and delta.

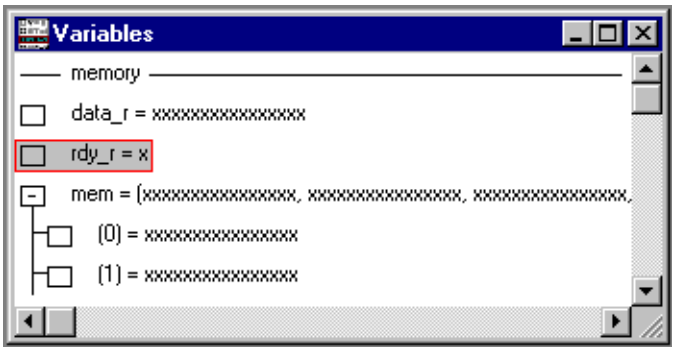
Structure window mouse actions

Use the mouse to:

- select a design entity by clicking on its name,
- collapse and expand levels by clicking on the plus and minus boxes.

Variables window

The Variables window lists the names of HDL items within the current process, followed by the current value(s) associated with each name. The pathname of the current process is displayed at the bottom of the window.



HDL items

The following HDL items for VHDL and Verilog are viewable within the Variables window:

- **VHDL**
  constants, generics, and variables - or -
- **Verilog**
  register variables

The names of any VHDL composite types (arrays and record types) are shown in a hierarchical fashion. Hierarchy also applies to Verilog vector memories. (Verilog vector registers do not have hierarchy because they are not internally represented as arrays.) To indicate hierarchy, each name is preceded by a box that contains a plus sign (+), a minus sign (-), or may be empty.

You can expand and contract the display to view the HDL items by clicking the mouse on the boxes at the left of the Window.

Variables window status bar

The Variables window status bar uses two fields:

- the current process,
- the simulation time and delta.

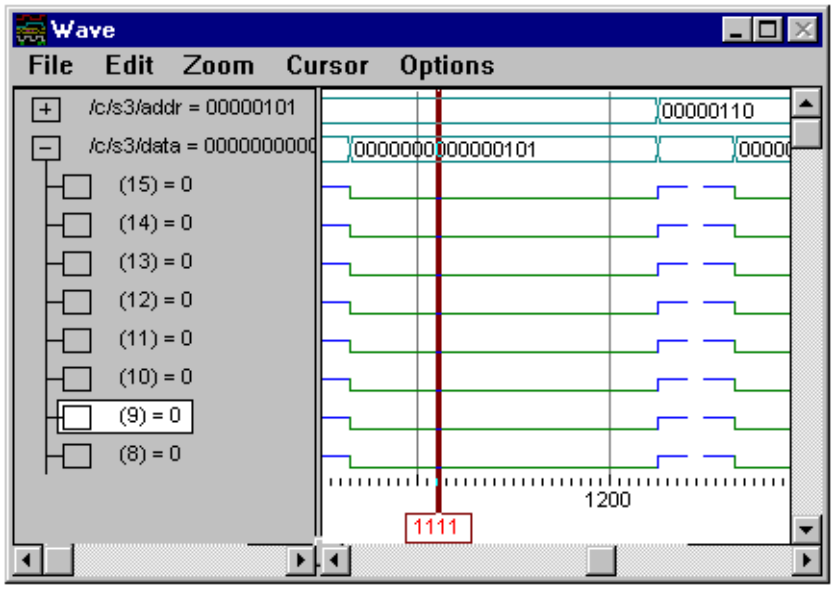
Variables window mouse actions

In the Variables window, you can use the left mouse button to:

- select an HDL item by clicking on its name,
- collapse and expand levels by clicking on the plus and minus boxes.

Wave window

The Wave window — like the List window — is another look into your simulation results. It has two *panels* or *windowpanes*; the left pane lists HDL item names and current values, while the right pane is a waveform display across a simulation time axis.



HDL items

The following HDL items for VHDL and Verilog are viewable within the Wave window:

- **VHDL** signals
- **Verilog** nets and register variables

The item values windowpane displays the name of the item that is associated with each waveform and the value of the item at the current cursor location. You can click on an array or record to expand it and show the values of its subelements. You can make multiple selections in the Wave window.

The data in the item values windowpane is very similar to the Signals window, except that values change dynamically whenever the cursor in the waveform windowpane is moved.

At the bottom of the waveform windowpane you can see a timeline, tick marks, and a readout of the cursor(s) position(s). To move a cursor, use the mouse. The time value at the cursor location is displayed at the bottom of the cursor.

From the Wave window you can:

- examine items as waveforms and see their values at any time during the simulation
- collapse and expand composites
- save the waveforms as a PostScript file
- make cursor measurements
- reformat items, change their display colors, and position them within the window
- zoom in or out to change the amount of simulation time shown in the window
- change the Wave window properties
- save the Wave window configuration

Wave window menu bar

File

Write PostScript File	write waveform simulation results to a file
Save Configuration	save the current window and HDL item formats

Edit

Cut	delete an HDL item and put it in the paste buffer
Copy	copy an HDL item and put it in the paste buffer
Paste	paste na HDL item
Delete All	delete all the HDL items in the Waveform window

Zoom

In 2X	show more detail and less simulation time
Out 2x	show more simulation time and less detail
Last	return to window view before last zoom
Range	specify range of simulation times to be shown

Cursor

Add	add a cursor to the display
Remove	remove the selected cursor

Options

Waveform Options	specify maximum length of HDL item name and snap distance
Signal Options	specify HDL item formatting

Wave window status bar

The Wave window status bar shows:

- the current simulation time and delta,
- and the simulation time interval visible in the waveform display.

Wave window mouse actions

You can perform different actions with the mouse depending on whether you are in the signal name/value panel (left windowpane) or in the waveform display panel (right windowpane).

In the right windowpane:

If you want to:	Use the:	And do this:
select an item	left button	click on the item's waveform - the nearest cursor (if any) is repositioned (delete cursors with Cursor > Remove from the Wave window menu)
move a cursor	left button	click on waveform in desired location - the nearest cursor is repositioned (add cursors with Cursor > Add from the Wave window menu)
zoom the display	right button	click and drag (rubberband) to zoom the display - the time expands to match the mouse movements

In the left windowpane:

If you want to:	Use the:	And do this:
select an item	left button	click on any item name
extend selection	<Shift> left button	click on any item above or below the current selection
add to selection	<Control> left button	click on any unselected item
subtract from selection	<Control> left button	click on any selected item
move item	left button	drag selected item to new position in the list
expand item levels	left button	click on a plus box
collapse item levels	left button	click on a minus box
call up Wave window menus	right button	click anywhere in the left windowpane

To resize the windowpanes:

Using the left mouse button, click and drag on the bar separating the windowpanes.

Adding HDL items to the Wave window

To add HDL items to the window, enter

```
wave <item_name> <item_name>
```

Separate the item names with a space.

You can add all the items in the current region with this command:

```
wave *
```

Or add all the items in the design with:

```
wave -r /*
```

You can also add items to the Wave window with the Signals menu. See "Adding HDL items to other windows" (p51).

Editing HDL items in the Wave window

Use the Wave window's menu bar **Edit** commands to edit the list of HDL items that appear in the item values pane. You can:

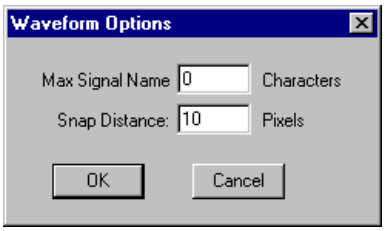
- **Cut** the selected item
- **Copy** and **Paste** the selected item
- or **Delete All** the items currently shown in the window

You can also reposition HDL items by dragging the item name to where you want it to appear.

Formatting the Wave window

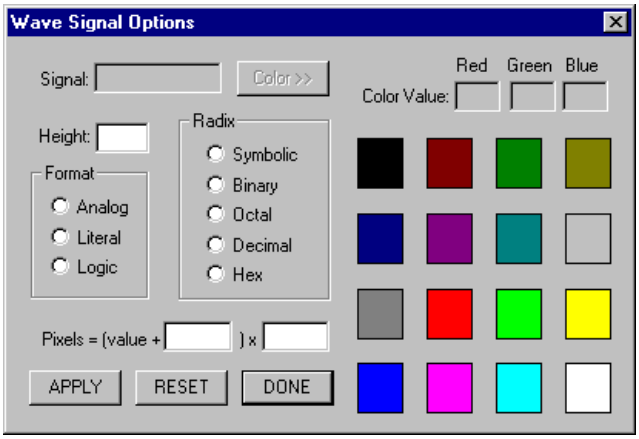
The Wave window menu bar **Options** menu allows you to format the display of HDL items and waveforms in both panes of the Wave window.

Use the Wave window's menu bar sequence: **Options > Waveform Options** to bring up the **Waveform Options** dialog box. In the dialog box you can specify how many characters are used for item names as well as the snap distance of the cursor.



- **Max Signal Name**
  Sets the HDL item name width. This is especially useful for items that have a long pathname. Choose a maximum name width setting, say 10 characters, and then item pathnames longer than 10 characters are truncated. All truncations take place at the slash boundary so you will never see a partial item name. The default value for this field is 0, which means to display the full path. The default value can be changed with a *.ini* file setting.
- **Snap Distance**
  Specifies the distance the cursor needs to be placed from an HDL item edge to update the value displayed in the item value windowpane (a 0 specification turns off the snap).

Use the Wave window's menu bar sequence: **Options > Signal Options** to bring up the **Wave Signal Options** dialog box. This option allows you to change the waveform format, radix and individual color.

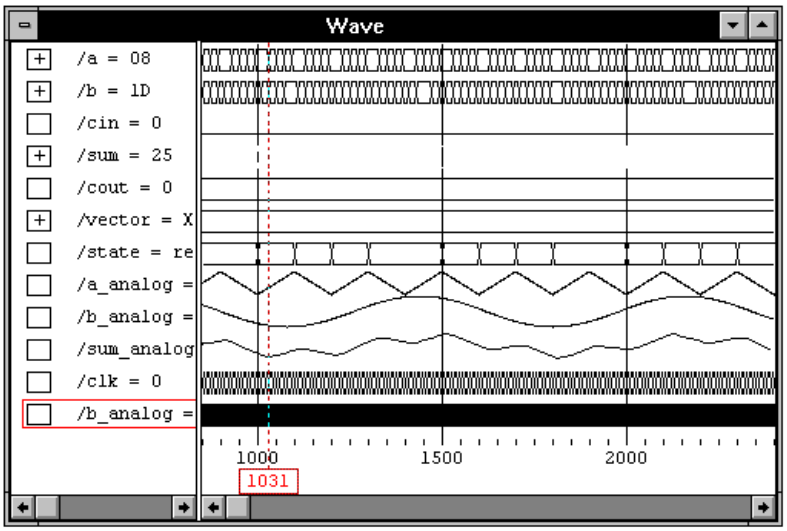


- **Signal:**
  Shows the current HDL item name.
- **Color**
  This command button expands the dialog box to the size shown in the illustration. Color formatting can apply to any HDL item. When you choose one of the colors shown, the **Color Value** shows you the color's red, green, and blue values.
- **Height:**
  This specification (in pixels) controls the spacing between HDL items in the Wave window.
- **Format**
  Depending on the HDL item's type, one or more formats will be available for formatting the display. See the *Analog, Literal and Logic Formatting* sections below for more detail.

Tip:

You can keep this dialog box open and select other HDL item in the wave window and change their colors, height and display format.

Analog formatting



- **Analog**
  Integer and floating point HDL items can be formatted as analog in the Wave window. Model*Sim* then plots the item value on the screen as shown above for **a_analog**, **b_analog**, and **sum_analog**. Compare the first **b_analog** item (formatted as analog) with the same item at the bottom of the display (formatted as literal).
- **Pixels =**
  These two text boxes can only be used with HDL items formatted as analog. They let you customize the Wave window display to make it easier to use. In the first text box, you specify an offset value that will be added to the item value before it is displayed. Use the second text box to set a scaling factor.

Literal formatting

If you choose **Literal** formatting, you can also specify the radix of the HDL item:

- symbolic (useful for enumerated types)
- binary

- octal
- decimal
- hexadecimal.

See **vector** in the illustration for an example of symbolic.

Logic Formatting

Choose **Logic** to format a HDL item display in logic representation (1, 0, X, etc.).

Zooming the Wave window

You can change the amount of time shown in the waveform display windowpane of the Wave window with the mouse or with menu commands.

Zoom in on an area of the display by holding down the **<right mouse button>** and dragging through the area of the display you want expanded.

You can also use these **Zoom** menu bar commands:

- **Full**
  Redraws the display to show the entire simulation from time 0 to the current simulation time.
- **In 2X**
  Zooms in by a factor of two, increasing the resolution and decreasing the visible range horizontally, cropping the view on the right. The starting time is held static.
- **Out 2X**
  Zooms out by a factor of two, decreasing the resolution and increasing the visible range horizontally, extending the view on the right. The starting time is held static.
- **Last**
  Restores the display to where it was before the last zoom command.
- **Range**
  Brings up a pop-up menu that allows you to enter the beginning and ending times for a range of time units to be displayed.

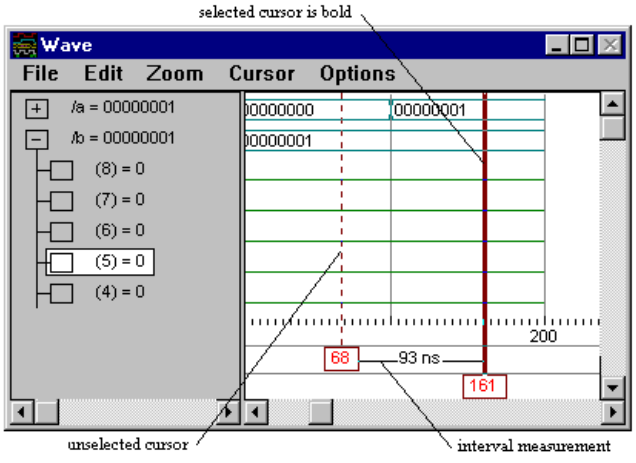
Another method of zooming is with the function keys:

Key	Action
<F4>	full zoom
<F5>	zoom in
<F6>	zoom out
<F7>	last zoom

For additional Model*Sim* shortcuts see: "Keyboard shortcuts" (p76)

Using the Wave window cursors

You can add or remove cursors with **Cursor > Add | Remove** menu command. Up to five cursors can be present at the same time. Cursors are displayed with a time box showing the precise simulation time at the bottom. When you have more than one cursor, each time box appears in a separate track at the bottom of the display.



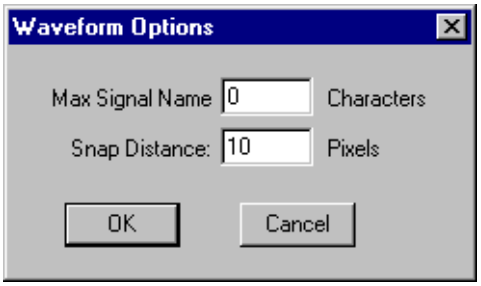
When the Wave window is first drawn, there is one cursor in it at time zero. Clicking anywhere in the waveform display brings that cursor to the mouse location.

When you add a cursor, it is drawn in the middle of the display. Once you have more than one cursor, Model*Sim* adds a delta measurement showing the time difference between the two cursor positions. The selected cursor is drawn as a solid line; all other cursors are drawn with dotted lines.

Tip:

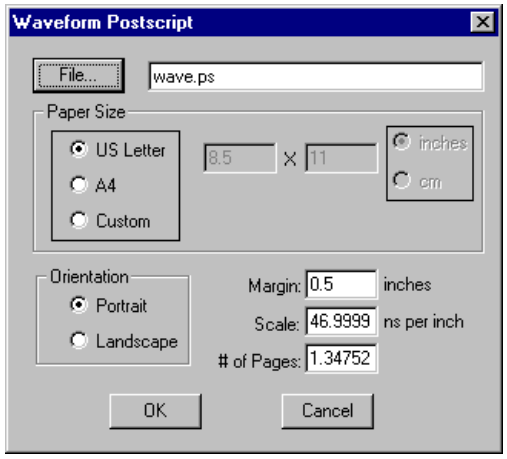
If you click in the waveform display, the cursor closest to the mouse position is selected and then moved to the mouse position. Another way to position multiple cursors is to use the mouse in the time box tracks at the bottom of the display. Clicking anywhere in a track selects that cursor and brings it to the mouse position.

The cursors are designed to snap to the closest HDL item edge to the left. You can control the snap distance from the dialog box available with the **Options > Waveform Options** command.



Writing a PostScript file

The Wave window's menu bar command, **File > Write PostScript File,** saves a history of all the HDL items in the Wave window. The item names appear to the left of the waveforms in the file.



Use the Waveform Postscript dialog box to change paper size and orientation, or scale the output to meet your needs. Note that the Margin, Scale, or Number of Pages specifications are interactive. If you change one of these, tab to or click in another box to see the updated specs.

The PostScript file includes a file named *vsim.ps* that is included with Model*Sim*. The file is "included" in the PostScript output from the waveform display and it sets the default fonts, spacing, and print header and footer. If you want to change any of the PostScript defaults you can do it by making changes in this file. Note that you should copy the file before making your changes so that you can save the original.

Saving the Wave window configuration

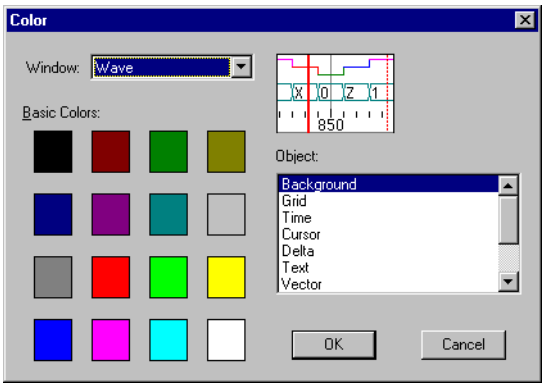
If you choose **File > Save Configuration** , you will get a dialog box which lets you save your Wave window configuration for use in a future simulation. This generates a macro file that contains "wave" commands to rebuild the waveform as it currently appears.

Customizing Model*Sim* windows

All Model*Sim* windows can be moved and sized with Microsoft Windows commands and procedures. The Window menu combines typical Microsoft Windows commands with commands for customizing Model*Sim* windows. This section details the **Colors** and **Fonts** menu selections.

Changing window colors

The **Window > Colors** menu bar command returns the **Color** dialog box.



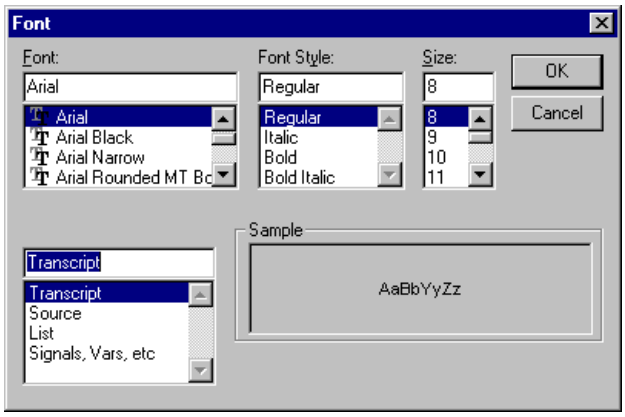
In the color dialog box, you can specify a color for each type of object in each of the Model*Sim* windows. For example, in the illustration, the Wave window is selected. And in that window, you can set the color of objects like the background, the cursors, text, etc. The dialog box also contains a miniature version of the Model*Sim* window which will be updated as you make your color selections.

Make these selections to change window colors:

- **Window:**
  Each of the Model*Sim* windows is available in this drop-down list box.
- **Object:**
  All the objects for the selected window are shown in this list box.
- **Basic Colors:**
  Use the palette to set the color for the selected object.

Changing window fonts

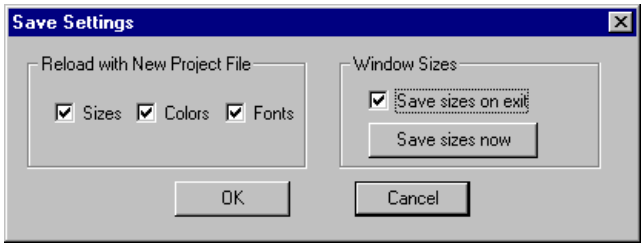
The **Window > Fonts** menu bar command brings up the **Font** dialog box.



In the **Font** dialog box, first you select the Model*Sim* windows in which you want to change fonts. The Transcript, Source, and List windows can be selected separately. The other six windows (Signals, Variables, Process, Structure, Dataflow and Wave) are grouped. Once you've chosen the windows, you can specify fonts with the typical list boxes. Note that monospaced fonts are best for the Source and List windows.

Saving your window settings

The **Window > Settings** menu bar command brings up the **Save Settings** dialog box.



Use the **Save sizes on exit** check box to save your Model*Sim* window sizes and positions. Then after starting a simulation, the **Window > Restore All** menu bar command will change the window icons back the way you left them when you last used Model*Sim*.

Depending on the task you were performing with Model*Sim*, you might have different window sizes, positions, fonts, and colors with each *modelsim.ini* project file you define. If you want Model*Sim* to change its window settings to match a new project file, use the check boxes in **Reload with New Project File**.

Keyboard shortcuts

The following keyboard shortcuts are available in Model*Sim* PE:

Key	Action
<Page Up>	scroll up one screen
<Page Down>	scroll down one screen
<Ctrl><left arrow>	scroll left one screen
<Ctrl><right arrow>	scroll right one screen
<Ctrl><Home>	move to beginning of window
<Ctrl><End>	move to end of window
<Ctrl><up arrow>	scroll up one line
<Ctrl><down arrow>	scroll down one line
<Shift><left arrow>	scroll left one column
<Shift><right arrow>	scroll right one column
<F4>	full zoom (Wave window only)
<F5>	zoom in (Wave window only)
<F6>	zoom out (Wave window only)
<F7>	last zoom (Wave window only)
<F9>	run simulation
<F10>	continue simulation
<F11>	single-step
<F12>	single-step over function call
<Ctrl> C	break (interrupt simulation run or macro execution)
<Esc>	cancels a dialog box

Several keyboard shortcuts exist for the Transcript window only. See "Transcript window keyboard actions" (p33) for more information.

4 - Tutorial: Using Model*Sim* PE

Chapter contents

After you complete the "Tutorial setup" (p78),
choose the lessons appropriate for your simulator version:

PLUS, and VHDL lessons

Basic VHDL simulation 79

Debugging a VHDL design 93

PLUS, and VLOG lesson

Basic Verilog simulation 105

PLUS lesson

Mixed VHDL/Verilog simulation 120

PLUS, VHDL, and VLOG practice

Learning more about ModelSim's windows. 127

Assumptions

We assume that you are familiar with the use of the Microsoft Windows operating system. If this is not the case, we recommend that you work through the tutorials provided with Microsoft Windows before using Model*Sim*.

We also assume that you have a working knowledge of HDL design. Although Model*Sim* is an excellent tool to use while learning HDL concepts and practices, this guide is not written to support that goal.

Additional details for VHDL, Verilog, and mixed VHDL/Verilog simulation can be found in the *ModelSim PE/PLUS Reference Manual*. See "Where to find our documentation" (p15).

Tutorial setup

Before we turn you loose with Model*Sim*, let's set up your application windows and save their settings so that if you are interrupted and restart, you will find the windows like you left them. In this setup you will learn how to start the application, and save your window settings.

Step 1.

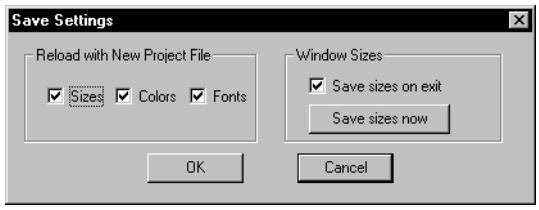
Start by invoking Model*Sim*: Double-click the Model*Sim* icon. The Model*Sim* application window will come up with the Transcript window showing. Maximize the Model*Sim* application window to fill your screen.

Step 2.

If you cannot see the tool bar or the status bar, use the **Window > Tool Bar** and **Window > Status Bar** menu sequences to make them visible.

Step 3.

Use the **Window > Settings** menu sequence and check all of the boxes indicated in the Save Settings dialog box shown below. Now whenever you exit Model*Sim*, your window settings will be saved.



Click **OK**. You can now exit, or continue on to "Basic VHDL simulation" (p79).

Note:

In the exercises, just type what's shown in this guide and then use the <RETURN> or <ENTER> key to complete the line.

Also, if you are using the PDF version of *Getting Started* you can copy the command text from the PDF file and paste it onto Model*Sim*'s command line.

Basic VHDL simulation

The goals for the first lesson are:

- create a library
- create a new project file
- compile a VHDL file
- start the simulator
- use the RUN command
- list some signals
- use the waveform display
- force the value of a signal
- single-step through a simulation run
- set a breakpoint
- save the listing from the List window to a file
- quit the simulator

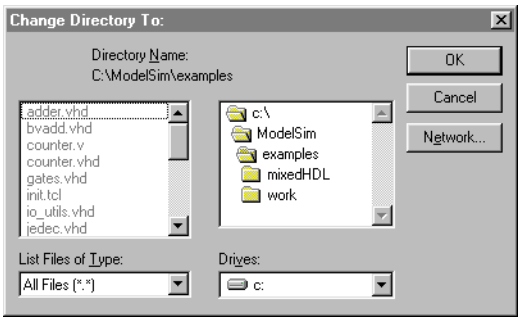
Step 1.

For this exercise you need to change directories to locate the tutorial files. To do this, pull down the **File** menu and select **Directory**.

This brings up the Change Directory To dialog box. You can navigate among the directories and view their contents in the files scroll box. The directory your looking for, *\examples*, is located within the directory that contains *modelsim.exe*, the Model*Sim* executable.

Step 2.

Double-click on the *examples* entry in the directories list box to move within the *\examples* directory. Click **OK**.



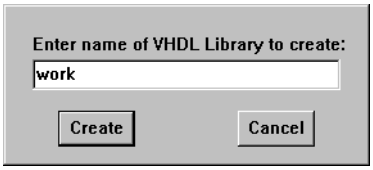
If you click Cancel, the dialog box is closed and you remain in the same working directory as before. You must click the **OK** button to actually change to a different directory.

Using the **File > Directory** menu command is the same as entering the **cd** (change directory) command at the Model*Sim* prompt in the Transcript window. In fact, the following command is echoed in the Transcript window:

```
cd C:\modelsim\examples
```

Step 3.

Before you can compile a VHDL source file, you need to create a design library to hold the compilation results. Pull down the **Library** menu and select **New**. This brings up a dialog box that prompts you to enter the name of the new VHDL library.



Enter the name **work** and click **Create**.

The command is echoed in the Transcript window:

```
vlib work
```

This creates a VHDL library named work under the current directory. Even though this command creates a DOS directory called *work*, it is important to realize that it is more than just a directory. You must use only the **Library > New** option or a **vlib** Model*Sim* command to create a new Model*Sim* library. Do not use the DOS mkdir command or Microsoft Windows to create the directory!

Note:

If this tutorial has already been run on your system, a message will report that the *work* subdirectory already exists in the current directory. You can ignore the message and proceed.

Step 4.

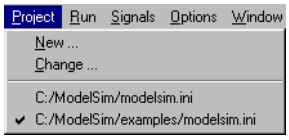
Now you need to create a new project file in your current working directory. This project file will remember things about your environment (such as window sizes and positions, etc.), so that next time you work in this directory, Model*Sim* will automatically reload your environment for you. To create a new project, select **Project > New**. Type in *modelsim.ini*, which is the name Model*Sim* expects to find when you come back to this project later.

Note:

A project file will already exist if someone has done this exercise before. Again, Model*Sim* will just give you a message that you can safely ignore.

Step 5.

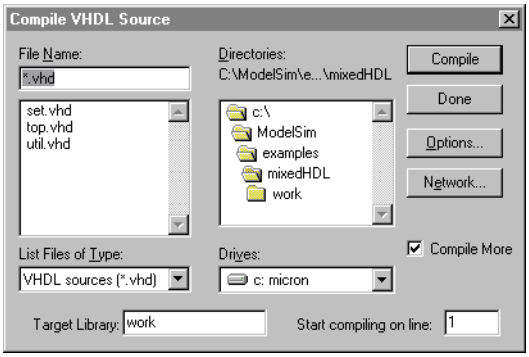
Pull down the **Project** menu again and notice that the file */examples/modelsim.ini* is checked, indicating that Model*Sim* is using that project file. If you wanted, you could change back to the original project file by simply selecting it from the menu, which would move the check mark to it - but don't do that.



Step 6.

Next, compile the VHDL source file named *counter.vhd*. To do this, pull down the **File** menu and select **Compile VHDL**.

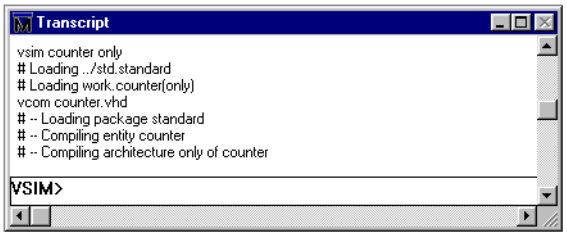
This brings up the Compile VHDL Source dialog box. For now, click **Done**.



Step 7.

Press the **VCOM** button on the tool bar. The same dialog box appears. Now select the VHDL source file for the counter, named *counter.vhd,* and then click **Compile**.

When you click **Compile**, the VHDL compiler is invoked and generates the messages shown below unless it encounters errors.

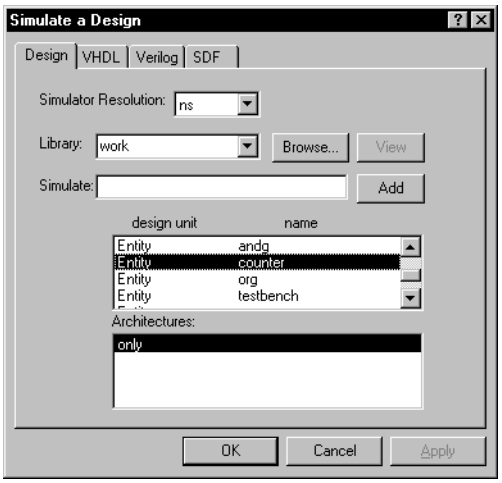


During compilation, the Model*Sim* prompt is not displayed. You will know that the compiler is finished when the Model*Sim* prompt returns to the Transcript window command line.

Step 8.

After the VHDL source file is compiled, close the dialog box by selecting **Done**. Now start the simulator by pulling down the **File** menu and selecting **Simulate**.

This brings up the Simulate a Design dialog box, where you select the library and the top-level design unit to simulate. You can also select the simulation time resolution. But select **Cancel** for now.



See "The Simulate a Design dialog box" (p28) for more detail about this dialog box.

Step 9.

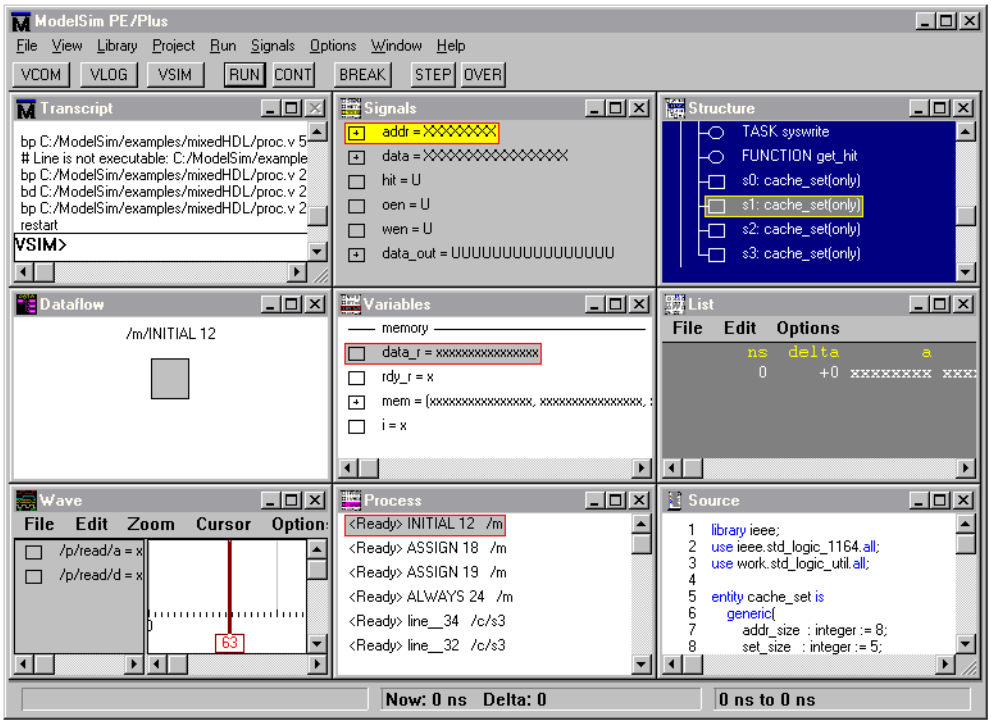
Click on the **VSIM** button in the tool bar. This time select the entity named **counter** to be simulated. When you select the entity name, the names of any associated architectures are displayed in the scrollbox at the bottom of the dialog. The counter only has one architecture, which is named only. Click **OK** to accept the settings.

The remaining Model*Sim* windows are now available for viewing within the original application window.

Step 10.

Use the **View > All** menu sequence to open all of the Model*Sim* windows. If you are the first person to use this tutorial, the windows will cascade. The windows may be in some other

arrangement if the tutorial has already been used. For now, select **Window > Tile Vertically**. The order of the windows might be different on your screen, but you should be able to see all nine as in the illustration.



Note that the Transcript window has echoed “view *” for your window command.

Step 11.

At this point, let’s put signals in both the List and the Wave windows. Type:

```
list *
```

and then:

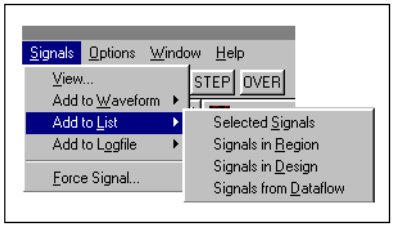
```
wave *
```

Notice that no matter which Model*Sim* window is active, anything you type is automatically entered on the Transcript window command line.

The commands you typed made signals appear in the List and Wave windows. If you desired to, you could list or view only specified signals by entering the signal names after the command instead of the asterisk. (When you do that, separate the names with spaces.)

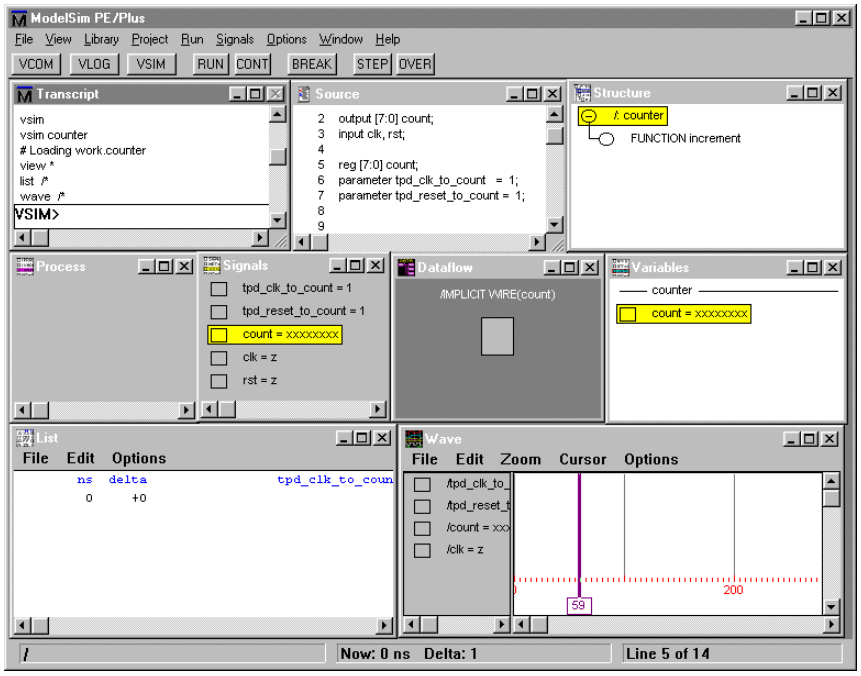
Step 12.

An alternative would be to use the menu bar. If you use the Signals menu, you will see several options.



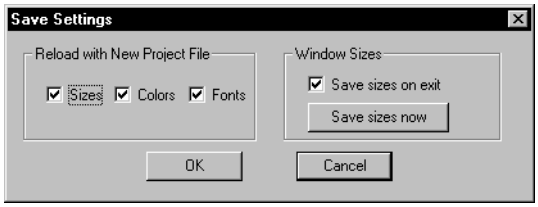
Step 13.

Before we go on, let's customize the window positions and sizes. Select **Window > Tile Vertically**. Make the Transcript, Source, and Structure windows about equal size across the top tier. Then put the Process, Signals, Dataflow, and Variables windows in the second row. Now if you make the List and Wave windows twice as big, you will be using the entire screen again - probably something like the illustration.



Step 14.

Use the **Window > Settings** menu sequence to be sure all your changes will be saved. Make sure these boxes are checked: **Sizes**, **Colors**, **Fonts** and **Save sizes on exit**.



Step 15.

Now let's apply some stimulus to the clock input of our circuit. Type:

```
force clk 1 50, 0 100 -repeat 100
```

This **force** command means:

- Force clk to the value “1” at 50 ns (nanoseconds) after the current time,
- then to “0” 100 ns from now, and
- repeat this cycle every 100 ns.

Step 16.

Select **RUN** on the tool bar. This causes the simulation run for the default simulation length, which is 100 ns. (The default simulation run length can be modified with **Options > Simulation Options** from the menu bar.)

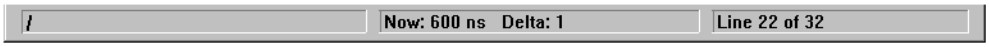
Other ways of entering a **run** command include typing it in the Transcript window, or selecting an option from the Run menu: **Run <default_run_length>, Run Forever, Step, Step Over, Cont** or **Break**. See "ModelSim menu bar" (p24) for more information about **run** options.

Step 17.

Next type:

```
run 500
```

Now that you have run the simulation for 600 nanoseconds, look at the status bar at the bottom of the Model*Sim* application window. It should look something like this illustration.



Step 18.

If your status bar is different, it is because the status bar changes with the active window. Click in another Model*Sim* window and examine the status bar again.

Step 19.

Let’s get back to running the simulator.

You can run the simulator for a long time by entering a large number of time units. You can also make the simulator run to a specific time with a command like this:

```
run @ <timestep>
```

Use the **run** command with @ to run to time 5000:

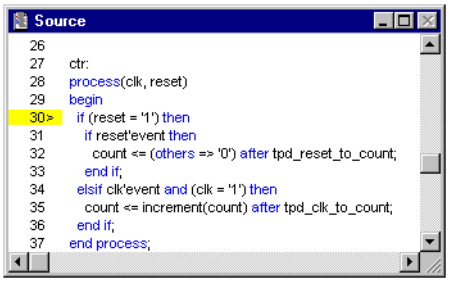
```
run @5000
```

Step 20.

To see the effect of the break feature, you are now going to start a long simulation run and then click the **BREAK** button during the run to stop the simulator. On the menu bar, select **Run** > **Run Forever**. (The **run -a** command produces the same effect.)

Step 21.

To stop the simulator while it is running, click the **BREAK** button. This updates the Source window; an arrow will be pointing to the next VHDL statement to be executed.

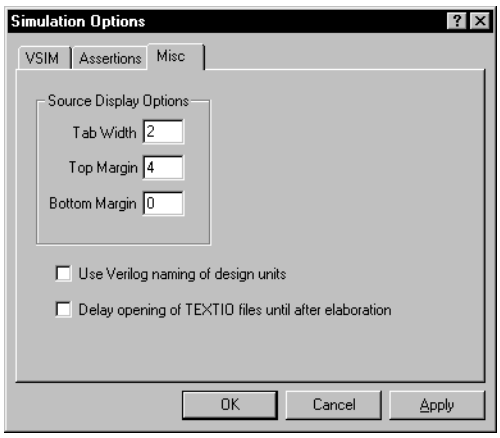


Note:

Your window won't look like the illustration above. Your simulation will probably stop at a different point.

Step 22.

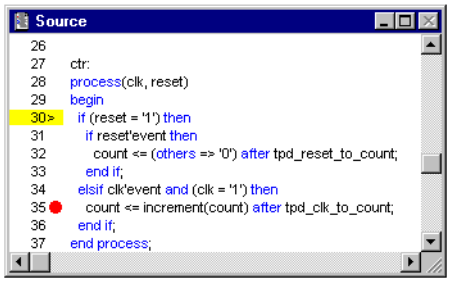
If your Source window is hard to read because the standard tabs are too big for the window size, change the tabs with the **Options > Simulation Options** menu sequence. Select the **Misc** tab in the **Simulation Options** dialog box, then change the **Tab Width** to 2 characters, leaving the other options unchanged. Click **OK**, then **Yes**, to save your changes.



Step 23.

Next, set a breakpoint on line 35, which has a call to the VHDL function named increment. To do this, move the pointer to the Source window and find line 35. Use the vertical scroll bar if necessary.

Double-click on line 35 to set the breakpoint. You will see a dot next to the line number when the breakpoint is set. A breakpoint can be toggled on and off by double-clicking on the line.



Step 24.

Select **CONT** (continue) on the tool bar to resume the run that you interrupted. When the simulator hits the breakpoint, it is indicated by highlighting in the Source window and by a message in the Transcript window.

Step 25.

Examine the current value of the signal count by entering the command:

```
examine count
```

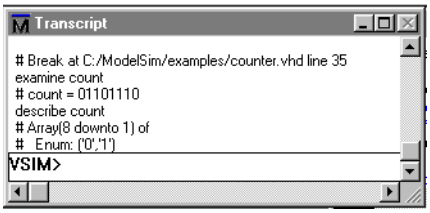
You can also see this information in the Signals window.

Step 26.

You can find out what kind of variable or signal count is by using the Model*Sim* command **describe**. To see this, enter:

```
describe count
```

The system should report (in the Transcript window) that count is an 8-bit array of an enumerated type:



Step 27.

Click **OVER** (for Step Over) in the tool bar to step over this instruction. If you had clicked STEP instead, the simulation would have entered the increment function called in line 35.

Step 28.

Re-examine the value of count and notice that its value has not changed. Even though the signal count has been assigned a new value, the counter has not yet been incremented. This is because a signal assignment does not take effect until after any specified propagation delays (specified using an after clause in its VHDL signal assignment statement).

Step 29.

Again, select **CONT** to resume the run.

Step 30.

On your own, try setting and deleting breakpoints at different points in the *counter.vhd* source file, and using the STEP and OVER buttons until you feel comfortable with these commands.

Step 31.

When you are ready, you can record the simulation values by writing the contents of the Model*Sim* List window to a file (CAUTION, this is optional, this could be a huge file). To do this, enter the following command in the Transcript window:

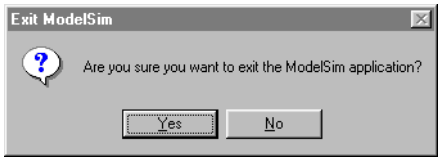
```
write list counter.lst
```

Alternately you can activate the List window (by clicking anywhere in it) and then choose **File > Save Listing**. The listing will resemble the illustration below.

```
      ns  delta    count clk reset
       0     +0 00000000   0     0
      50     +0 00000000   1     0
      55     +0 00000001   1     0
     100     +0 00000001   0     0
     150     +0 00000001   1     0
     155     +0 00000010   1     0
   .
   .
   .
   62555     +0 01110010   1     0
   62600     +0 01110010   0     0
   62650     +0 01110010   1     0
   62655     +0 00000000   1     0
   62700     +0 00000000   0     0
   62750     +0 00000000   1     0
```

Step 32.

Now quit the simulator. You can type **exit**, or you can choose **File > Exit ModelSim** with the Transcript window active. Model*Sim* then returns a dialog box.



At this point you can confirm the action with the **Yes** button. If you want to skip the dialog box, enter the command:

```
exit -force
```

Debugging a VHDL design

The goals for this lesson are:

- use a VHDL testbench (a *testbench* is a VHDL architecture that instantiates the VHDL design units to be tested, provides simulation stimuli, and checks the results)
- map a logical library name to an actual library
- change the default run length
- recognize assertion messages in the command window
- change the assertion break level
- restart the simulation run using the **restart** command
- change the value of a variable
- use a strobe to trigger lines in the Model*Sim* List window
- list signals in different radixes in the List window

Step 1.

Restart Model*Sim* if necessary, then change to the tutorial file directory by pulling down the **File** menu and selecting **Directory**.

This brings up the Change Directory To dialog box. Double-click the subdirectory *examples* and click **OK**.

Step 2.

To create the new library, pull down the **Library** menu and select **New**. This brings up a dialog box asking you the name of library you want to create.

Step 3.

Enter the name **lib2** and click **Create**. This creates a new library directory under the current working directory. In the Model*Sim* window, you will see the following command echoed:

```
vlib lib2
```

Note:

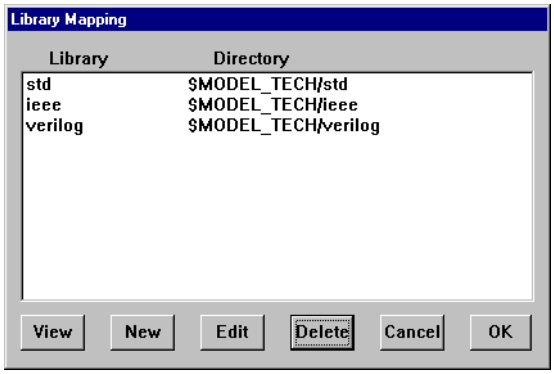
If this tutorial has already been run on your system, a message will report that the *lib2* subdirectory already exists in the current directory. You can ignore the message and proceed.

Step 4.

The next step is to set your work library to be the **lib2** library you just created. In the last tutorial, you created a work library called **work**, but now you will map the **work** library to **lib2**. The old **work** library is still there, but it will not be referenced by Model*Sim* if a logical mapping for **work** exists.

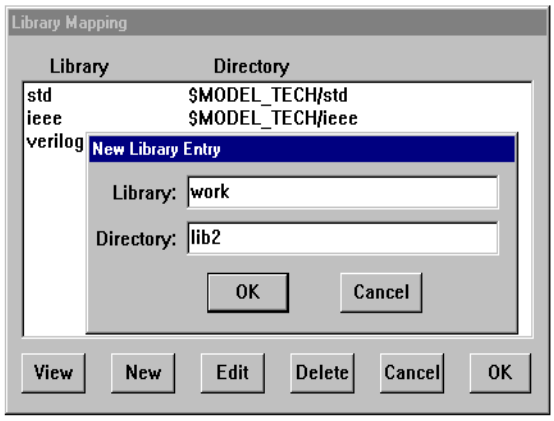
Step 5.

Now pull down the **Library** menu and select **Mapping**. A dialog box like the one shown below will appear.



Step 6.

Now click on the **New** button. This brings up another dialog box where you can type in the desired mapping.



In this case, type in **work** for the library name and **lib2** for the directory name. This tells Model*Sim* that whenever it sees a reference to the **work** library, use the **lib2** library instead. Click on **OK** to accept the new library mapping.

Step 7.

None of the changes you've made to the library mapping take effect until you click on OK in the Library Mapping dialog box. This way you can add, edit, and delete mappings as much as you like and then discard the changes by clicking Cancel. Go ahead and click **OK** now.

Step 8.

Next, compile the design units for this tutorial:

- *gates.vhd*
- *adder.vhd*
- *testadd.vhd*

You can do this four different ways. One, by selecting Compile VHDL from the File menu and compiling the source files one-by-one using the Compile VHDL Source dialog box. Two, you could access the same dialog box with the VCOM button on the tool bar. Three, you can enter the following sequence of commands at the Model*Sim* prompt:

```
vcom gates.vhd
vcom adder.vhd
vcom testadd.vhd
```

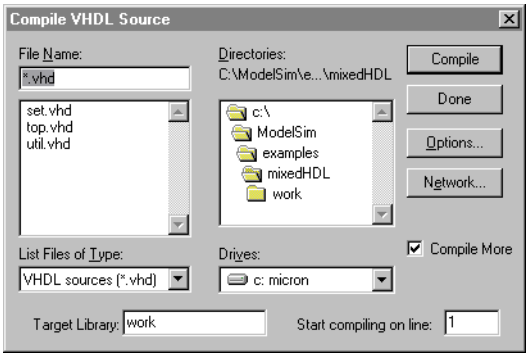
And four, you could compile multiple files using a single **vcom** command, as in the following example:

```
vcom gates.vhd adder.vhd testadd.vhd
```

Let's use the tool bar and the Compile VHDL Source dialogue box. Press the **VCOM** button on the tool bar.

Step 9.

Once you have the Compile VHDL Source dialog box on the screen, note the Compile More check box.



When Compile More is checked, the dialog box stays open until you click on Done. If the box is unchecked, each time you compile, the dialog box is closed. This is handy on smaller screens because then you can see the compiler messages in the Transcript window.

Check **Compile More** now.

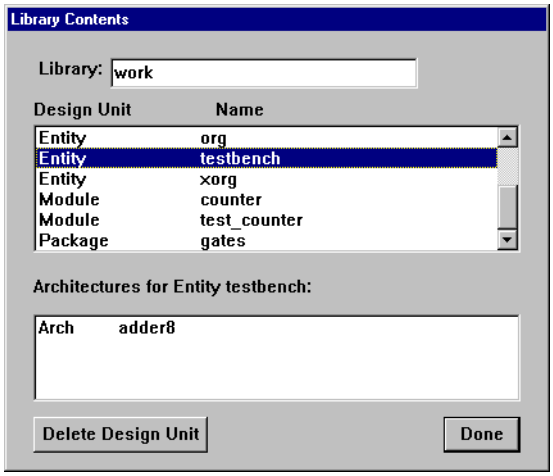
Now compile the source files. Note that the order you do this is important because *testadd.vhd* uses *adder.vhd* and *adder.vhd* in turn uses *gates.vhd*, so compile them in this order:

- *gates.vhd*
- *adder.vhd*
- *testadd.vhd*

When you're finished, click **Done**.

Step 10.

You can confirm that these design units have been compiled into the library by pulling down the **Library** menu and selecting **Contents**. This brings up a dialog box that displays the names of all the design units in the library.

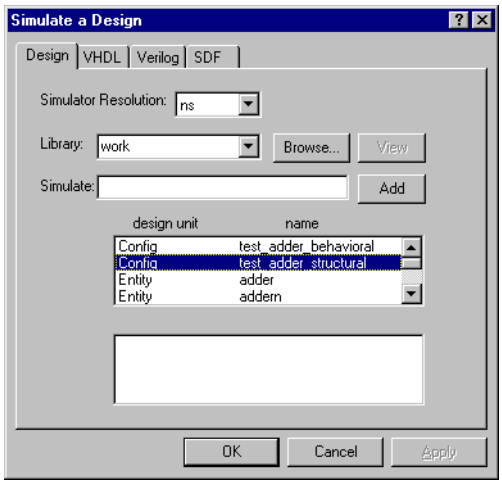


Step 11.

Scroll through the list of configurations and entities to see what has been compiled into the **work** library. Choose an entity and click on it, and all of its architectures will be shown in the lower list box. When you are done, click **Done** to close the window.

Step 12.

Now start the simulator by pulling down the **File** menu and selecting **Simulate** (or use the VSIM tool bar button). However you choose to invoke the simulator, the Simulate a Design dialog box is displayed.



Step 13.

Select the Design tab and perform the following steps:

- make sure that the simulator resolution is **ns**
- look in the design unit scrollbox and select the configuration named **test_adder_structural** (because this design unit is a configuration, no architectures are displayed in the architecture scrollbox at the bottom of the dialog)
- click **OK** to accept the settings

Now open the other eight Model*Sim* windows with the **View > All** menu sequence. The windows will appear in the sizes and positions in which we left them in the previous lesson. We'll first use the Source and Structure windows to examine the design.

Step 14.

Double-click in the title bar of the Structure window to maximize it. Model*Sim* automatically generates this view from your VHDL code. Click on the minus sign in the box for **testbench(adder8)** at the top level.

Now you can explore the design by collapsing and expanding the levels of hierarchy shown in the window. A level of hierarchy is created by each component instantiation, generate statement, block statement, and package.

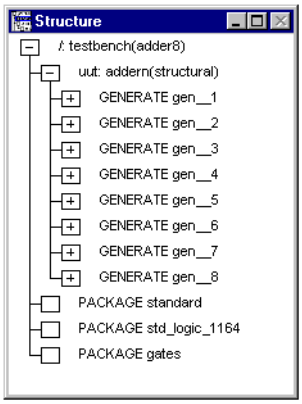
The plus sign means other levels exist below it. The minus sign is used on fully-expanded items. An empty box has no lower levels.

Step 15.

When you finish exploring, return the Structure window to its original size by clicking on the restore button (the one in the upper right - with small overlapping windows).

Step 16.

Now click on a name (as opposed to a box) in the Structure window. Note that as you click on parts of the design in the Structure window, both the Source and Signals windows change to show the correct source file contents and the signals in it.



Step 17.

Select **testbench(adder8)** in the Structure window by clicking on the name before going on.

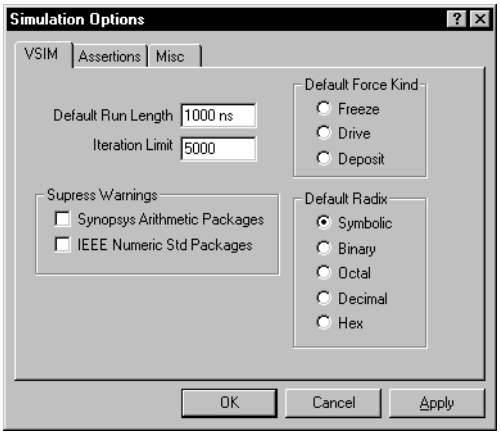
Step 18.

Put signals in the List and Wave windows by typing:

```
list *
wave *
```

Step 19.

Change the default simulation run length; choose **Options > Simulation Options**.



On the **VSIM** tab, change the **Default Run Length** to 1000. Do not change the other settings for now.

Click **OK,** then **Yes,** to accept the new settings.

Step 20.

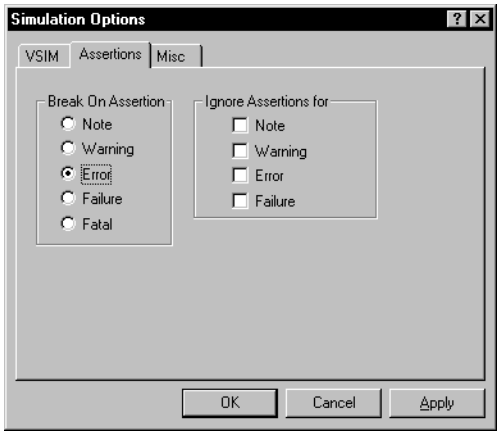
Next, you will run the simulator. Click **RUN** on the Model*Sim* tool bar.

A message in the Transcript window will notify you that there was an assertion error.

Let's find out what's wrong. Perform the following steps to track down what caused the assertion message.

Step 21.

Select **Options > Simulation Options**.



Step 22.

Under the **Assertions** tab, change the selection for **Break on Assertion** to **Error** and click **OK**, then **Yes**. This will cause the simulator to stop at the VHDL statement after the assertion is displayed.

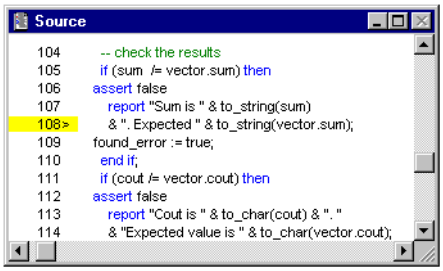
Step 23.

To restart the simulation, pull down the **File** menu and select **Restart**.

For now, leave all the options as they are and select **OK**.

Step 24.

Next, click **RUN** on the tool bar. Notice that the arrow in the Source window is pointing to the statement after the assertion.


Source
104 -- check the results
105 if (sum /= vector.sum) then
106 assert false
107 report "Sum is " & to_string(sum)
108> & ". Expected " & to_string(vector.sum);
109 found_error := true;
110 end if;
111 if (cout /= vector.cout) then
112 assert false
113 report "Cout is " & to_char(cout) & ". "
114 & "Expected value is " & to_char(vector.cout);

Step 25.

The assertion occurred because the signal sum does not equal vector.sum, which is derived from the sum field in test_patterns. The sum of the inputs a, b, and cin should be equal to the output sum. The error is in the test vectors.

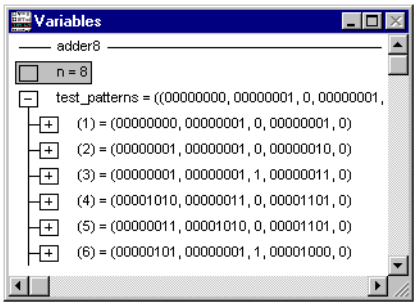
Step 26.

To correct this error, you need to restart the simulation and modify the initial value of the test vectors. But before you restart, determine which element of the test_patterns array needs to be modified.

From the assertion error message, you can see that the expected pattern was wrong: 0101 plus 0001 plus 1 (the carry) should be 0111, not 1000.

Step 27.

Now look at the **test_patterns** array with the Variables window. Use the plus boxes to expand the elements and find the element that has the sum field value of 00001000. It is the sixth line in the list, so you can refer to that element as test_patterns(6), and the sum field of that can be referred to as test_patterns(6).sum.



Step 28.

Select **Restart** from the **File** menu. Restart resets the values to those that were set when you first entered the simulator. Click **OK**.

Step 29.

Now we can change the value of the sum record; type:

```
change test_patterns(6).sum 00000111
```

Note that this is a temporary edit, you must use your text editor to permanently change the source code.

Step 30.

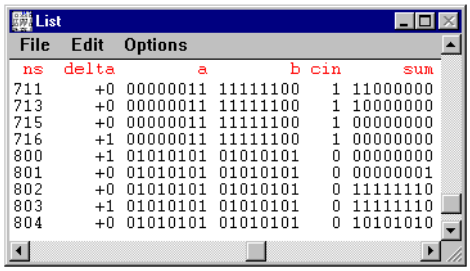
Select **RUN** on the tool bar. At this point, the simulation will run without any errors.

Step 31.

Your next action will be to add a, b, and sum to the List window and specify that their values are to be displayed using the decimal radix. To do this, enter the following command:

```
list -decimal a b sum
```

Now you should have a List window with contents like the illustration.



Note that since you used the **list -decimal** command, the a, b, and sum signals appear twice in the List window, once in binary and then again in decimal format.

Step 32.

This brings you to the end of this lesson. Feel free to experiment further with the menu system if you wish. (You may also want keep this simulation running for the following keyboard and mouse practice session.) When you are ready to end the simulation session, quit Model*Sim* without saving data by selecting **Exit ModelSim** from the **File** menu.

You can also do this by typing the following command (the same as **exit -force**) :

```
exit -f
```

Basic Verilog simulation

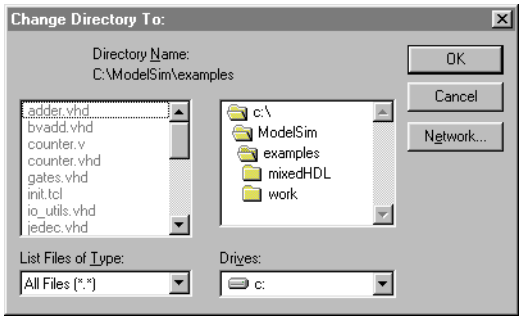
The goals for this lesson are:

- compile a Verilog design
- examine the hierarchy of the design
- list signals in the design
- change list attributes
- set a breakpoint

Step 1.

First, you need to change directory to the folder where the tutorial files are located. To do this, pull down the **File** menu and select **Directory**.

This brings up the Change Directory To dialog box with which you can browse the file system of your PC and choose directories via the directories list scrollbox.



Notice the dialog box may not default to the directory where Model*Sim* is installed. You may need to browse to locate the Model*Sim* install directory and the tutorial files. The tutorial folder is called *examples* and should be in the same directory as *modelsim.exe*, the Model*Sim* executable.

Step 2.

Move inside the *examples* folder by double-clicking the *examples* entry in the directories list box. Click **OK**

If you click Cancel, the dialog box is closed and you remain in the working directory you were in before. You must click **OK** to actually change to a new directory.

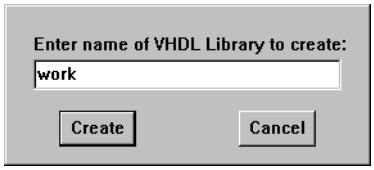
Using the **File > Directory** menu command is the same as entering the change directory (**cd**) command at the Model*Sim* prompt in the Transcript window. In fact, notice how your change directory dialog box actions were echoed in the Transcript window:

```
cd C:\modelsim\examples
```

Step 3.

Pull down the **Library** menu and select **New**. This brings up a dialog box where you can enter the name of your new library.

Before you can compile a Verilog design, you need to create a library to store the compilation results. If you are only familiar with interpreted Verilog simulators such as Cadence Verilog XL this will be a new idea for you. Because Model*Sim* is a compiled Verilog, it requires a target design library for the compilation. Model*Sim* can compile both VHDL and Verilog code into the same library if desired.



Enter the name **work** and click **Create**.

Notice how the command is echoed in the Transcript window:

```
vlib work
```

This creates a design library named **work** in the current working directory. Even though this appears to be a DOS directory, it is in fact much more. You must use only the **Library > New** option (or a **vlib** Model*Sim* command) to create a new Model*Sim* library. Do not use the DOS mkdir (or an equivalent Microsoft Windows operation) to create a design library directory!

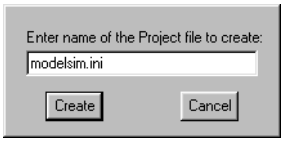
Note:

If this tutorial has already been run on your system, or you've just completed Lesson 1, a message will report that the **work** subdirectory already exists in the current directory. You may safely ignore this message and proceed.

Step 4.

Next you need to create a new project file in your current working directory. This project file will be used by Model*Sim* to remember your environment settings (such as window sizes and positions, etc.) so the next time you work from this directory Model*Sim* can reload the environment for you.

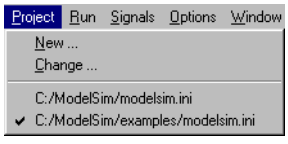
To create a new project file, select **Project > New**. Type in *modelsim.ini*, which is the name Model*Sim* expects to find when you return to this project later.



A project file may already exist in the current directory if someone has previously run this tutorial. Again, Model*Sim* may give you a warning message to that effect, which you can safely ignore.

Step 5.

Pull down the **Project** menu again and notice that the file *examples/modelsim.ini* is checked, indicating it is currently in use by Model*Sim*. If you wanted, you could change back to the previous project file by simply selecting it from the menu. This would move the check mark - but don't do that now.



Next, you need to compile the Verilog design.

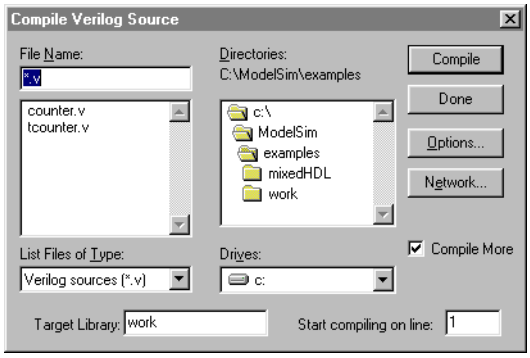
The example design we'll be using consists of two Verilog source files, each containing a unique module. The file *counter.v* contains a module called counter that implements a simple 8-bit binary up-counter. The other file (*tcounter.v*) is a testbench module (test_counter) used to verify counter. Under simulation you will see that these two files are configured hierarchically with a single instance (instance name dut) of module counter instantiated by the testbench. You'll get a chance to look at the structure of this code later.

For now, you need to compile both files into your **work** design library. There are two ways to compile design units with Model*Sim*'s graphic interface: from the menu bar, or from the tool bar. We'll look at both options.

Step 6.

Pull down the **File** menu and select **Compile Verilog**.

This brings up the Compile Verilog Source dialog box. Notice this dialog box lists all files in the current directory with a *.v* suffix which is a useful convention for naming Verilog source files. You could select a file here and click Compile but don't do that just yet. Instead click **Done**.



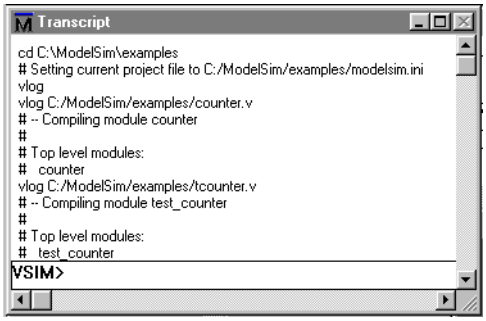
Step 7.

Click the **VLOG** menu button.

The same Compile Verilog Source dialog box appears. Notice in the compile dialog box a selector box called **Compile More**, select it now. When this is selected, the dialog box remains active after you click Compile so you can compile more than one source file.

Now select any Verilog source file and click **Compile**.

When you click Compile, VLOG (the Verilog compiler) is invoked and should generate output messages similar to these:



Notice that the order in which you compile the two Verilog modules is not important. This may again seem strange to Verilog XL users who understand the possible problems of interface checking between design units, or compiler directive inheritance. Model*Sim* defers such checks until the design is loaded by VSIM (the HDL simulator), so it doesn't matter if you choose to compile *counter.v* before or after *tcounter.v*.

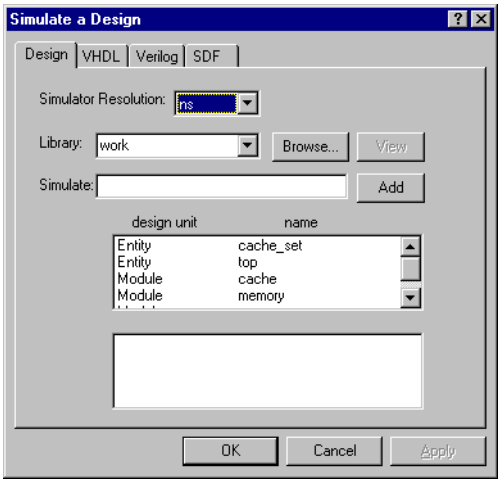
Select the next Verilog file and click **Compile**.

During compilation, the Model*Sim* prompt is not displayed. When the compiler finishes the prompt returns. When all files have been compiled, click **Done**.

Now that the design files have been compiled, you are ready to simulate. Before you move on however, it is interesting to pull down the **Library > Contents** menu. This shows the a list of the current contents of the active library (**work** in this case). You should see the two modules of this design: counter, and test_counter. One option available to you here is to delete one or more of the design modules from the library - but don't do that now. Instead click **Done**.

Step 8.

Now let’s start the simulation. You can either pull down the **File** menu and select **Simulate**, or click the menu button labeled **VSIM**.



Either action brings up the Simulate a Design dialog box. This dialog box has four tabs to select simulation options: Design, VHDL, Verilog, and SDF.

At this point, you can accept the defaults in each case. Click the **Design** tab, make sure **work** is the targeted library and select the top-most module in the design **test_counter**. If you choose to simulate only the counter module you will have a very short and uneventful simulation run since test_counter contains the clock generator and other stimulus necessary for counter to run.

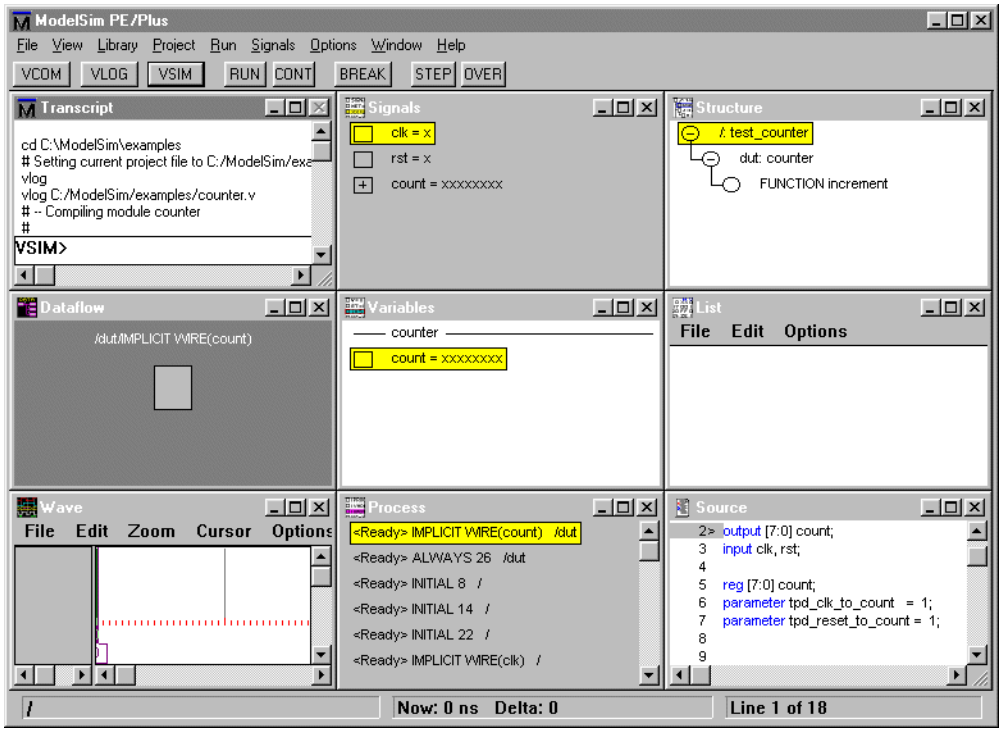
See "The Simulate a Design dialog box" (p28) more information on the **Simulate a Design** dialog box.

Step 9.

Click **OK** to invoke the simulator.

The remaining Model*Sim* windows are now available for viewing within the original application window.

Use the **View > All** menu sequence to open all of the Model*Sim* windows. If you are the first person to use this tutorial, the windows will cascade. The windows may be in some other arrangement if the tutorial has already been used. In any case, make the **Window > Tile Vertically** menu selection. This should arrange the nine windows similar to the following illustration.

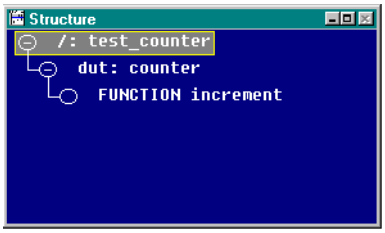


The order of the windows may be different on your machine but you should see all nine. See "ModelSim PE Graphic Interface" (p21) for a detailed description of each window.

Although you have invoked the simulator, and runtime checking of the code has occurred, the simulation is halted at time zero waiting for your command.

Step 10.

Click inside the Structure window.



Notice how this window describes the hierarchical structure of the design. In the illustration the Structure window shows three hierarchical levels: test_counter, counter and the function called increment.

You can navigate within the hierarchy by clicking on a line within the window.

Step 11.

Click on **FUNCTION increment** and notice how other VSIM windows are automatically updated as appropriate.

Specifically, the Source window displays the Verilog code at whichever hierarchical level you selected in the Structure window. Using the Structure window in this way is analogous to scoping commands in interpreted Verilogs. For now, make sure the **test_counter** module is showing in the Source window by clicking on the top line in the Structure window.

Step 12.

Before we run the simulation, let's put some signals in the List and Wave windows. You can do this either by typing a command, or using the pull down menus. Let's do one of each. First, type:

```
wave *
```

This causes the Wave window to display the signals clk, rst and count. The ***** is a wildcard and here it means all signals at the current hierarchical level. (Typing **wave -r *** would have displayed all signals throughout the design hierarchy.) In a moment, when you run the simulation you will see the Wave window update with signal values.

Step 13.

Secondly, pull down **Signals > Add to List > Signals in Design**.

This causes the List window to display the signals clk, rst and count as well as the current simulation time. (The same effect could be achieved by typing **list -r *** .)

If you desired, you could invoke these commands to display individual signals by specifying the signal names instead of the ***** wildcard or by selecting **Signals > Add to List > Selected Signals** from the pulldown menus.

Notice that whenever you type a command in Model*Sim* it appears in the Transcript window. This is true regardless of which window you have currently active.

Step 14.

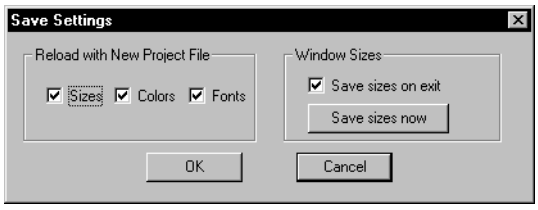
At this point why not take a moment to customize your window positions and sizes. Try to dedicate the most screen-space to the Transcript, Wave, Source and List windows.

Note:

As you become familiar with the Model*Sim* environment you will find yourself using some windows more than others. You can easily make more room for yourself in the application window by minimizing some of the Model*Sim* windows you don't currently need.

Step 15.

When you are happy with your work, pull down **Window > Settings** to be sure your changes will be saved into the *modelsim.ini* file. Make your selections as in this illustration.



Step 16.

Click **RUN** on the toolbar.

This causes the simulation to run for the default simulation length, 100ns. (The default simulation run length can be changed with Options > Simulation Options from the menu bar.)

Instead of clicking the RUN button you could also type the **run** command.
To try it now, type:

```
run 500
```

Now the simulation has run for a total of 600ns (the default 100ns plus the 500 you just asked for). See the status bar at the bottom of the Model*Sim* window. It should show something like this illustration.

If your status bar is different it is because the status bar changes depending upon the active Model*Sim* window. Try selecting a different window and look at the status bar again.

Step 17.

The last command you typed (**run** 500) caused the simulation to advance for 500ns. You can also advance simulation to a specific time. Type:

```
run @ 3000
```

This advances the simulation to time 3000ns. Note that the simulation actually ran for 2400ns (3000 - 600). Also note that the Transcript window echoes the results specified in the test_counter module; the time echoed is not intended to match the time shown on the Model*Sim* status bar.

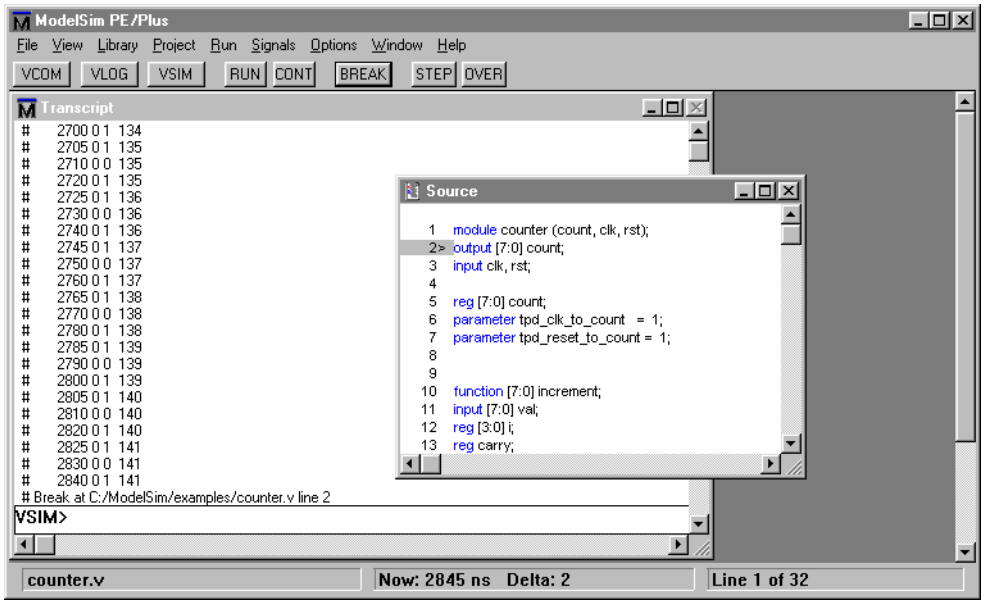
Step 18.

If you wish, you can also use the **run** command to simply run, with no predetermined stop time. Pull down **Run > Run Forever** from the menu.

Step 19.

You may stop the simulator while it is running by a simple click. Do so now by clicking the **BREAK** button on the toolbar.

This halts the simulation and updates the Source window, which now features a > (greater-than sign) next to the line of source code about to be executed. See the illustration.


ModelSim PE/Plus
File View Library Project Run Signals Options Window Help
VCOM VLOG VSIM RUN CONT BREAK STEP OVER
Transcript
2700 0 1 134
2705 0 1 135
2710 0 0 135
2720 0 1 135
2725 0 1 136
2730 0 0 136
2740 0 1 136
2745 0 1 137
2750 0 0 137
2760 0 1 137
2765 0 1 138
2770 0 0 138
2780 0 1 138
2785 0 1 139
2790 0 0 139
2800 0 1 139
2805 0 1 140
2810 0 0 140
2820 0 1 140
2825 0 1 141
2830 0 0 141
2840 0 1 141
Break at C:/ModelSim/examples/counter.v line 2
VSIM>
Source
1 module counter (count, clk, rst);
2> output [7:0] count;
3 input clk, rst;
4
5 reg [7:0] count;
6 parameter tpd_clk_to_count = 1;
7 parameter tpd_reset_to_count = 1;
8
9
10 function [7:0] increment;
11 input [7:0] val;
12 reg [3:0] i;
13 reg carry;
counter.v
Now: 2845 ns Delta: 2
Line 1 of 32

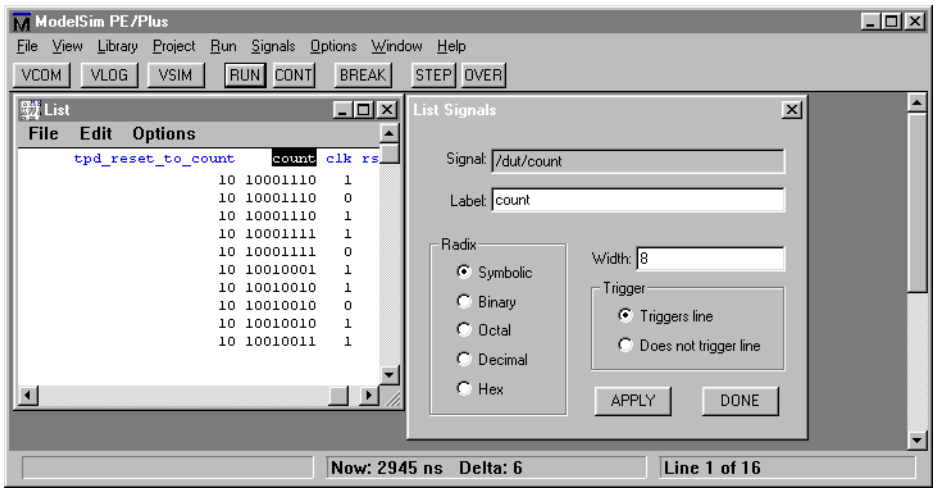
Your windows won't look exactly like the illustration because your simulation very likely stopped at a different point. Also, we are showing only two windows in this illustration.

Next we'll take a brief look at some interactive debug features of the Model*Sim* environment. To start with, let's see what we can do about the way the List window presents its data.

Step 20.

In the List window select the signal named **count**.

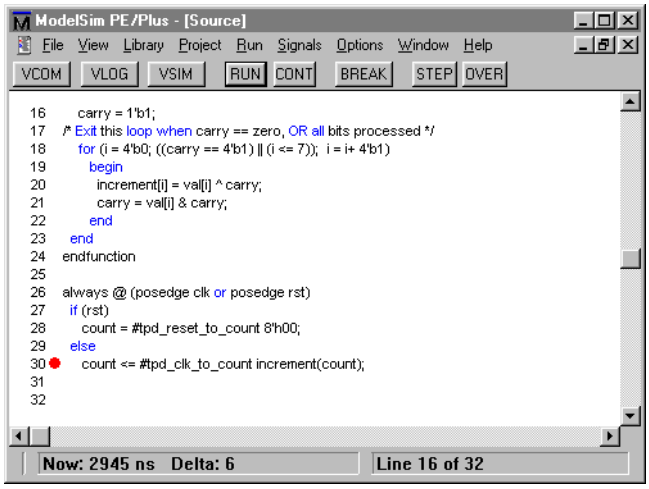
From the List window menu bar, pull down **Options > Signal Options** and select a display radix of **Decimal**. Click **DONE**. This causes the List window output to change; the count signal is now listed in decimal rather than the default binary.



Step 21.

Model*Sim* includes an extensive breakpoint feature. Make sure the second line (dut:counter) in the Structure window is selected, then set a breakpoint at line 30 in the Source window. To set the breakpoint, simply move the cursor to the Source window and scroll the window to display line 30, and double-click on line 30.

You will see a dot appear next to the line number. This indicates that a breakpoint has been set for that line. Double-clicking line 30 once more would remove the breakpoint - but don't do that now.


ModelSim PE/Plus - [Source]
File View Library Project Run Signals Options Window Help
VCOM VLOG VSIM RUN CONT BREAK STEP OVER
16 carry = 1'b1;
17 /* Exit this loop when carry == zero, OR all bits processed */
18 for (i = 4'b0; ((carry == 4'b1) || (i <= 7)); i = i+ 4'b1)
19 begin
20 increment[i] = val[i] ^ carry;
21 carry = val[i] & carry;
22 end
23 end
24 endfunction
25
26 always @ (posedge clk or posedge rst)
27 if (rst)
28 count = #tpd_reset_to_count 8'h00;
29 else
30 count <= #tpd_clk_to_count increment(count);
31
32
Now: 2945 ns Delta: 6
Line 16 of 32

Step 22.

Select **CONT** from the toolbar to resume execution of the simulation. When the simulation hits the breakpoint, it stops running, highlights the Source window and issues a message in the Transcript window.

Step 23.

Typically when a breakpoint is reached you will be interested in one or more signal values. There are a few ways to determine this. You can:

- look at the values shown in the Signals window, or
- examine the current value of the signal count by typing:

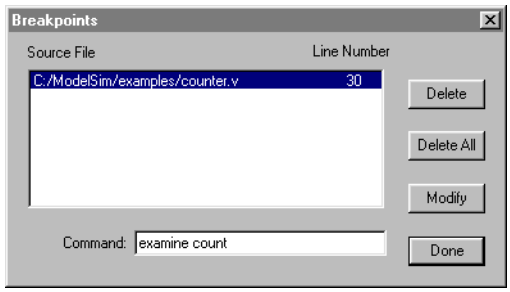
```
examine count
```

Perhaps more usefully, you can ask Model*Sim* to automatically issue the **examine** command when the breakpoint is reached.

Step 24.

Pull down **Options > Edit Breakpoints** from the main menu bar.

This brings up the Breakpoints dialog box. This dialog box should look like the illustration.



Notice that the breakpoint we entered is displayed. Highlight the breakpoint for line 30 by clicking on it. Now click the **Modify** button in this dialog box. The Command field becomes active and you can type into it:

```
examine count
```

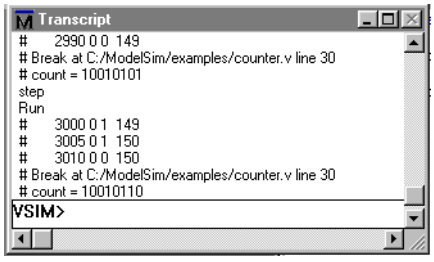
This command will be executed every time the breakpoint on line 30 is reached.

Click **Done**, answer **Yes** to the new dialog box that asks you to confirm your action, and click **Done** once more to close the Breakpoints dialog box.

Step 25.

Select **CONT** from the toolbar.

When the breakpoint on line 30 is triggered, the simulator stops running as before but now you can see output from the automatic examine count command in the Transcript window.



Step 26.

Click **OVER** on the toolbar.

This causes the simulator to step over the function call on line 30. The STEP button on the toolbar would have single-stepped the simulator, including each line of the increment function.

Step 27.

Experiment by yourself for a while; setting and clearing breakpoints as well as STEP'ing and OVER'ing function calls until you feel comfortable with the operation of these commands.

Step 28.

When you are ready, you can save the output of the VSIM List window if you like, by entering the following command (CAUTION, this file could be large, so this step is optional):

```
write list counter.lst
```

You could, of course, accomplish the same task by using the pulldown menus File > Save Listing.

Step 29.

You can quit the simulator now by typing exit or by choosing the **File > Exit ModelSim** menu bar sequence. You will be asked to confirm your decision to exit simulation by clicking the **Yes** button in a dialog box.

Alternatively, if you want to skip the confirmation dialog box type:

```
exit -force
```

Mixed VHDL/Verilog simulation

You must be using Model*Sim* PE/PLUS for this lesson.

Thanks to Model*Sim*'s single kernel simulator, the VHDL and Verilog simulation procedures you learned in the previous lessons apply directly to mixed VHDL/Verilog simulation.

The goals for this lesson are:

- compile multiple VHDL and Verilog files
- simulate a mixed VHDL and Verilog design
- list VHDL signals and Verilog nets and registers
- view the design in the Structure window
- view the HDL source code in the Source window

Step 1.

Restart Model*Sim*. Using the **File > Directory** menu sequence, and the **Change Directory To:** dialog box, move to the *\example\mixedHDL* directory (within the install directory).

In the Transcript window, Model*Sim* echoes the **cd** command.

Step 2.

Before you can compile an HDL source file, you need to create a new **work** library. Type this command in the Transcript window (or use the **Library > New** menu sequence):

```
vlib work
```

Note:

If this tutorial has already been run on your system, a message will report that the *work* subdirectory already exists in the current directory. You can ignore the message and proceed.

Step 3.

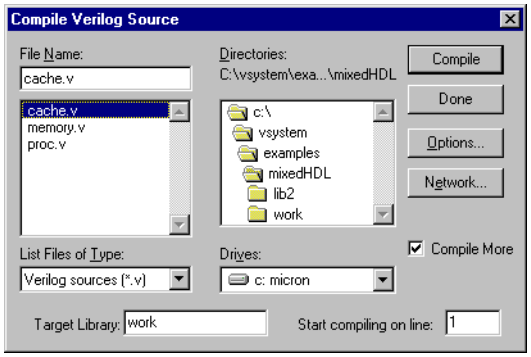
Now you need to create a new project file in your current working directory. Select **Project > New** and type in *modelsim.ini*.

Note:

A project file will already exist if someone has done this exercise before. Again, Model*Sim* will just give you a message that you can safely ignore.

Step 4.

Now you can compile the Verilog files using the **File > Compile_Verilog** menu sequence. Model*Sim* displays the Compile Verilog Source dialog box:



Compile *cache.v*, *memory.v* and *proc.v* in any order you choose. Note that a group of Verilog files can be compiled in any order. In a mixed VHDL/Verilog design, however, the Verilog files must be compiled before the VHDL files.

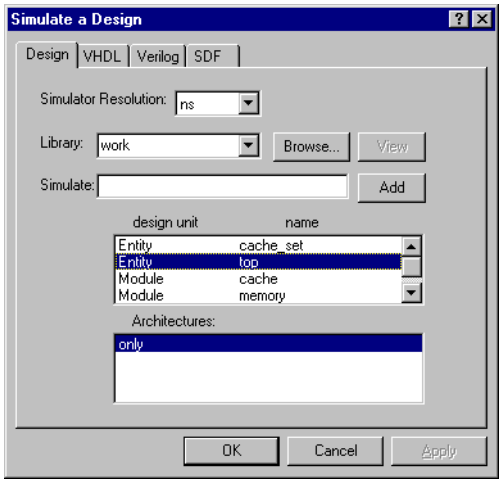
Step 5.

Depending on the design, the compile order of VHDL files can be very specific. In the case of this lesson, the file *top.vhd* must be compiled last. Compile the VHDL files from the Transcript window command line with this command:

```
vcom util.vhd set.vhd top.vhd
```

Step 6.

Begin simulation from the **VSIM** button on the tool bar. Invoke **vsim** with the name of the top-level design (**top**) from the Simulate a Design dialog box:



Step 7. From the menu bar select the **Window > Restore All** option. Arrange the windows so the List, Wave, Source and Structure windows are within view.

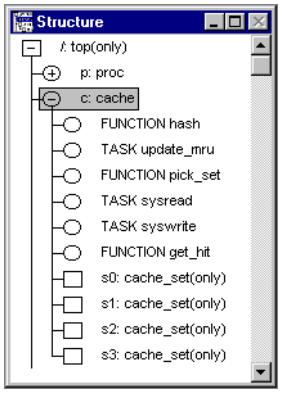
Step 8.

This time you will use the VSIM command line to add all of the HDL items in the region to the List and Wave windows:

```
list *
wave *
```

Step 9.

The Structure window is hierarchical. Notice the mixture of VHDL and Verilog in the design. VHDL levels are indicated by a square “prefix” in the Structure window, while Verilog levels are indicated by a circle “prefix.” Try expanding (+) and contracting (-) the structure layers. You’ll find Verilog modules have been instantiated by VHDL architectures and VHDL by Verilog.

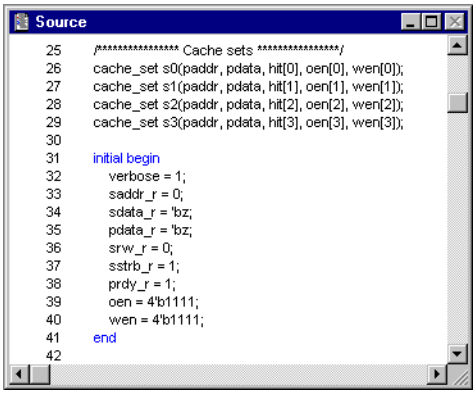


Let's take another look at the design.

Step 10.

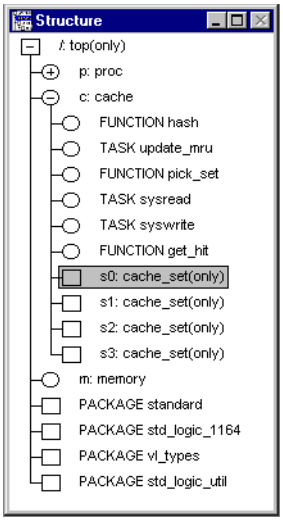
Click on the Verilog module, **c: cache**.

The source code for the Verilog module is now shown in the Source window. Scroll down to line #25. Note the declaration of cache_set; this is a VHDL entity instantiated within the Verilog file *cache.v*.

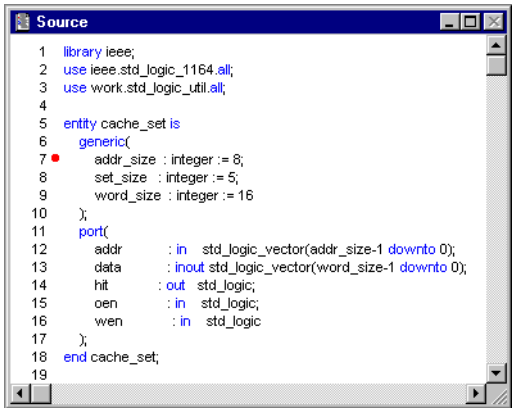

Source
25 /*************** Cache sets ***************/
26 cache_set s0(paddr, pdata, hit[0], oen[0], wen[0]);
27 cache_set s1(paddr, pdata, hit[1], oen[1], wen[1]);
28 cache_set s2(paddr, pdata, hit[2], oen[2], wen[2]);
29 cache_set s3(paddr, pdata, hit[3], oen[3], wen[3]);
30
31 initial begin
32 verbose = 1;
33 saddr_r = 0;
34 sdata_r = 'bz;
35 pdata_r = 'bz;
36 srw_r = 0;
37 sstrb_r = 1;
38 prdy_r = 1;
39 oen = 4'b1111;
40 wen = 4'b1111;
41 end
42

Step 11.

Click on the line "s0:cache_set(only)" in the Structure window.



The Source window now shows the VHDL code for the cache_set entity.



Step 12.

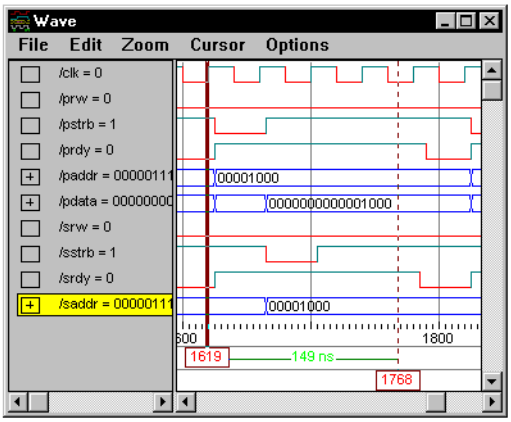
Try experimenting with some of the toolbar commands you've used in previous lessons: RUN, BREAK, CONT, STEP and OVER. Note that in this design, clk is already driven, so you won't need to use the **force** command.

Step 13.

Now let's get a Wave window view of the simulation; activate the Wave window.

When the Wave window is first drawn, there is one cursor in it at time zero. Clicking anywhere in the waveform display brings that cursor to the mouse location.

Up to ten cursors can be present at the same time. Cursors are displayed with a time box showing the precise simulation time at the bottom. When you have more than one cursor, each time box appears in a separate track at the bottom of the display.



Step 14.

Try adding or removing cursors with the **Cursor > Add | Remove** command.

When you add a cursor, it is drawn in the middle of the display. Once you have more than one cursor, VSIM adds a delta measurement showing the time difference between the two cursor

positions. The selected cursor is drawn as a solid line; all other cursors are drawn with dotted lines.

Step 15.

Click in the waveform display. Notice how the cursor closest to the mouse position is selected and then moved to the mouse position.

Another way to position multiple cursors is to use the mouse in the time box tracks at the bottom of the display. Clicking anywhere in a track selects that cursor and brings it to the mouse position.

The cursors are designed to snap to the closest wave edge to the left on the waveform that the mouse pointer is positioned over.

You can position a cursor without snapping by dragging in the area below the waveforms.

Step 16.

Experiment with using the cursors, scrolling, and zooming (see "Wave window mouse actions" (p63) for Wave window shortcuts).

Step 17.

When you’re done experimenting, quit the simulator by entering the command:

```
quit -f
```

Learning more about Model*Sim*'s windows

The more you use Model*Sim* for your own purposes, the better you will understand how it works; hence this unstructured practice session.

To begin your practice session simply start a simulation (you can use one of the previous lesson simulations if you wish). Once you're simulating, select a practice subject from the references below. When you're through practicing, end the simulation with one of the options you've learned in a previous lesson.

Dataflow window practice

- Tracing HDL items with the Dataflow window (p36)
- Dataflow window mouse actions (p36)

List window practice

- List window menu bar (p38)
- List window mouse actions (p39)
- Selecting HDL items to list (p40)
- Selecting List window triggering options (p40)
- Formatting a list (p41)
- Editing a list (p43)

Process window practice

- Process window mouse actions (p46)

Signals window practice

- Signals window mouse actions (p48)
- Selecting HDL item types to view (p48)
- Forcing HDL item values (p49)
- Adding HDL items to other windows (p51)

Source window practice

- Selecting the source file (p53)
- Editing breakpoints (p54)
- Examining values at breakpoints (p54)
- Customizing the Source window (p54)

Structure window practice

- Structure window mouse actions (p58)

Variables window practice

- Variables window mouse actions (p60)

Wave window practice

- Wave window menu bar (p62)
- Wave window mouse actions (p63)
- Adding HDL items to the Wave window (p65)
- Editing HDL items in the Wave window (p65)
- Formatting the Wave window (p65)
- Analog formatting (p68)
- Literal formatting (p68)
- Logic Formatting (p69)
- Zooming the Wave window (p69)
- Using the Wave window cursors (p70)

Keyboard practice

- Keyboard shortcuts (p76)

Continuing with Model*Sim* PE

More information on Model*Sim* commands, functions and techniques for use can be found in the following locations:

- **The Model*Sim* PE/PLUS Reference Manual, and**
- **the Tips and Techniques appendix of the Model*Sim* PE/PLUS Reference Manual.**

A - Help, Updates, and Licensing

Help

Technical Support for Mentor Graphics Customers

For customers who purchased from Mentor Graphics in North America, the support line number is 800-547-4303. The web site is http://supportnetweb.mentorg.com.

For customers who purchased from Mentor Graphics outside of North America, please contact your local support organization.

Technical Support for Model Technology Customers

For customer who purchased from Model Technology, please contact Model Technology via email: support@model.com. The telephone number for the support line is 503-641-1340 from 8:00 AM to 5:00 PM Pacific Time. Be sure to have your server hostID or hardware security key serial number handy. You can also use our web site for technical support: http://www.model.com.

Technical Support for Other Channels

For customers who purchased Model*Sim* as part of a bundled product from an OEM or VAR, support is available at the following:

- **Annapolis Microsystems**
  email: wftech@annapmicro.com
  telephone: 410-841-2514
  web site: http://www.annapmicro.com
- **Exemplar Logic**
  email: support@exemplar.com
  telephone: 510-789-3333
  web site: http://www.exemplar.com
- **Hewlett Packard EEsof**
  email: hpeesof_support@hp.com
  telephone: 1-800-HPEESOF (1-800-473-3763)
  web site: http://www.hp.com/go/hpeesof

- **Lucent Technologies**
  Bell Labs Design Automation
  email: info@blda.lucent.com
  telephone: 800-875-6590
  web site: http://www.bell-labs.com/org/blda
- **Synplicity**
  email: support@synplicity.com
  telephone: 408-617-6000
  web site: http://www.synplicity.com

Updates

Getting the latest version information

If you want the latest version information via email from Model Technology, make sure your email address is current in our customer database. If you think it is not, send email to license@model.com. Make the subject line read "register email address". Put your hostID or hardware security key number and your email address in the body of the message.

Getting the latest version

You can ftp the latest version of the software from the web site at ftp.model.com. Instructions are there as well. A valid license file from either Model Technology or Mentor Graphics is needed to uncompress the Model*Sim* EE files. A password from Model Technology is needed to uncompress the Model*Sim* PE files. Contact license@model.com if you are a current PE customer and need a password.

Model*Sim* PE Licensing

Starting with 4.6, Model*Sim* PE products use a programmable hardware security key. The "key code" for these devices consists of 20 hexadecimal digits, for example:

```
1234-5678-90ab-cdef-1234.
```

Embedded in the key code is: the serial number of the key (it is also printed on the third line on the back of the key); the product purchased (VHDL, VLOG, or PLUS); the maintenance

expiration date. The key code permits the use of any Model*Sim* PE software with a build date on or before the maintenance expiration date.

PE Maintenance Renewals

When maintenance is renewed, Model Technology will send a new key code that incorporates the new maintenance expiration date. If maintenance is not renewed, the key will still permit the use of any version of the software built before the maintenance expired.

Note:

PE users should protect the key. It is the most valuable part of your purchase. Model Technology will replace any defective key, but will not replace lost keys. Consider insuring the key for the purchase price of the product.

Entering Your Permanent Authorization Code

Make this menu selection when you start Model*Sim*: Help > About ModelSim > Authorization. (If your authorization has expired, Model*Sim* will prompt you with a dialog box.) Key in your new code and click Apply; the authorization information changes to reflect the current type of authorization and the new maintenance end date. Click Ok to complete the authorization.

When it is permanently programmed, your key will run all products of the correct type (VHDL, VLOG, or PLUS) with build dates before your maintenance expiration date. When your maintenance expires you can still run any product built up until that date. When you renew your maintenance, you will receive a new authorization code that will extend the date.

Getting help

The best thing is to email us a test case. Our email address is support@model.com. The next best thing is to copy/print the following form, fill it out, and fax it to us at 503-526-5473.

==========

Model Technology Fax Support Form

Your name:

Your company:

Your phone number:

Your FAX number:

Your email address:

Model*Sim* Version:
(Use the Help About dialog box with Windows; type **vcom** for workstations.)

Description of the problem (please include the exact wording of any error messages):

PC hardware security key serial number:

Host ID of licence server for workstations:

B - Resources

Appendix contents

Books 135

Organizations 137

Corporations & Consultants 138

Online resources 141

Books

Applications of VHDL to Circuit Design
Randolph E. Harr and Alec G. Stanculescu, Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 02061, USA. 1991.

Chip-Level Modeling with VHDL
James R. Armstrong, Prentice Hall, Englewood Cliffs, New Jersey 07632, USA. 1988.

The Designer's Guide to VHDL
Peter J. Ashenden, Morgan Kaufmann Publishers, Inc., 340 Pine Street, Sixth Floor, San Francisco, California 94104-3205, USA. 1995. (ISBN: 1-55860-270-4)

A Guide to VHDL Syntax
Jayaram Bhasker, Prentice Hall, Englewood Cliffs, New Jersey 07632, USA. 1995.

IEEE Standard VHDL Language Reference Manual
ANSI/IEEE Std 1076-1993. Published by The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, New York 10017-2394, USA. ISBN:1-55937-376-8, June 1994. To purchase the standard, write IEEE Customer Service, Hoes Lane, Tiscataway NJ 08855-1331, or phone 800-678-4333 (908-562-5420 from outside the U.S.), 908-981-9667 fax.

Practical Programming in Tcl and Tk
Brent Welch, Prentice Hall, Englewood Cliffs, New Jersey 07632, USA. 1995.

Tcl and the Tk Toolkit
John K. Ousterhout, published by Addison-Wesley Publishing Company, Inc. (ISBN 0-201-63337-X).

Verilog Language Reference Manual 2.0
Open Verilog International, 15466 Los Gatos Blvd., Suite 109-071, Los Gatos CA 95032 Phone: (408) 353-8899 , fax: (408) 353-8869 , email: ovi@netcom.com, Contact: Lynn Horobin

VHDL, 2nd Edition
Douglas L. Perry, McGraw-Hill, Inc., Professional Publishing Group, 11 West 19th Street, New York, New York 10011, USA. 1993.

VHDL: Analysis and Modeling of Digital Systems
Zainalabedin Navabi, Northeastern University, McGraw-Hill Publishing Company, College Division, 1221 Avenue of the Americas, New York, New York, 10020, USA. 1993.

VHDL Coding Styles and Methodologies
Ben Cohen, Kluwer Academic Publishers, Boston, Massachusetts, USA. 1995. (ISBN: 0-7923-9598-0)

VHDL: Hardware Description and Design
Roger Lipsett, Carl Schaefer, & Cary Ussery, Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 02061, USA. 1989.

VHDL Primer
Jayaram Bhasker, Prentice Hall, Englewood Cliffs, New Jersey 07632, USA. 1992.

Organizations

IEEE

The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, New York 10017-2394, USA. Fax:212-705-7453.

Open Verilog International

15466 Los Gatos Blvd., Suite 109-071, Los Gatos CA 95032
Phone: (408) 353-8899 , fax: (408) 353-8869 , email: ovi@netcom.com, web: http:// www.ovi.org Contact: Lynn Horobin The SDF mapping rules described in the Model Development Specification are based on version 2.1 of OVI's Standard Delay Format Specification. This document is available from Open Verilog International.

VHDL International

407 Chester Street, Menlo Park, CA 94025-3718, USA. Phone: 800-554-2550 or 415-329-0578, fax: 415-324-3150. A consortium of VHDL tool vendors and users that exists to promote the usage and standardization of VHDL. For information about VHDL International membership and mailings, contact Conference Management Services.

VHDL International Users' Forum

407 Chester Street, Menlo Park, CA 94025-3718, USA. Phone: 800-554-2550 or 415-329-0578, fax: 415-324-3150. The users' group within VHDL International. This group currently holds meetings twice a year in the spring and fall, which are attended by several hundred delegates. For information about VIUF meetings, membership, newsletters, and bulletin board services, contact Conference Management Services.

Corporations & Consultants

Automata Publishing Company
1072 South Saratoga-Sunnyvale Road, BLDG A107, San Jose CA 95129 USA. Phone: 800-8-VIPER1 or 408-255-0705, fax 408-253-7916, email: info@apco.com. PCI VHDL simulation models. Contact: Mona Singh.

CAST Computer Aided Software Technologies, Inc.
24 White Birch Drive, Pomona NY 10970 USA. Phone: 914-354-4945, Fax: 914-354-0325; email: info@cast-inc.com, web: http://www.cast-inc.com, VHDL services including model development, FPGA and ASIC design kits, training, simulation consulting. Contact: Newton Abdalla.

Comit Systems Inc.
1250 Oakmead Parkway, Suite 210, Sunnyvale CA 94088 USA. Phone: 408-988-2988; fax: 408-988-2133; email: bala@comit.com. FPGA design, VHDL modeling for embedded systems and related areas. Contact: Balachandran Sreekandath.

Daemon Modeling
834 SW Westwood Drive, Portland OR 97201 USA. Phone: 503-977-0554, email: 74051.3421@compuserve.com. Model development with emphasis on processor models. Contact: Daemon Anastas.

DP Consulting
9015 NE 139th Street, Kirkland WA 98034 USA. Phone: 206-821-9124. Model development. Contact: Richard Pado.

EDA Solutions Pty. Ltd.
Level 3, South Tower, 1-5 Railway Street Chatswood NSW 2067, Australia. Phone: 011 61 2 9413-4611, fax: 011 61 2 9413-4622, email: info@eda.com.au. Provides VHDL training, consulting services, and VHDL model development. Contact: Alain Legrand.

EEC Consultants, Inc.
3243 Springwood Drive, Clearwater FL 34621 USA. Phone: 813-786-2929. VHDL models. Contact: Kent Ulrich.

Hardi Electronics AB

Box 966, S220 09 Lund, Sweden. Phone: (46) 11-77-90; fax: (46) 13-15-99. Provides VHDL training, consulting services, and product distribution. Contact: Lars-Eric Lundgren.

LEDA SA

35 Avenue du Granier, 38240 Meylan, France. Phone: (33) 76-41-92-43. fax: (33) 76-41-92-44. Provides VHDL training, consulting services, and product distribution. Contact: Francis Sourbier.

Logic Modeling

19500 NW Gibbs Drive, Beaverton OR 97006 USA. Phone: 503-690-6900, fax: 503-690-6906. Provides modeling libraries.

Paolo Tabacco

Via Dei Berio 91, Rome Italy (00155). Phone: 39-6-22595-306; fax: 39-6-2280739; email: tabacco@space.it. Designs for boards and ASICs using Model*Sim* and Exemplar CORE in aerospace and commercials applications. Contact: Paolo Tabacco.

Papillon Research Corporation

52 Domino Drive, Concord MA 01742 USA. Phone: 508-371-9115; fax: 508-371-9175; email: info@papillonres.com; internet: http://www.papillonres.com/~papillon. Full product design including ASIC, FPGA, synthesis and modeling, analog, EMI and mechanical engineering. Contact: Dave Matthews.

Proxy Modeling

1580 Washington Blvd., Fremont CA 94539 USA. Phone: 510-440-VHDL. Provides modeling and consulting. Contact Keith G. Irwin or Pam G. Rissmann.

Saros Technology Limited

St. Michaels House, 94 High Street, Wallingford, Oxon OX10 OBW, United Kingdom. Phone: 011-44-1491-837787, fax: 011-44-1491-837477, email: carey@saros.co.uk, internet: http://www.saros.co.uk. Providers of Model*Sim* support in the UK. Developers of the VHDL editor, Turbo Writer. Contact: Carey Sayer

Seva Technologies Inc.
200 Brown Road, Suite 103, Fremont, CA 94539 USA.
Phone: 510-249-9085, fax: 510-249-9082, email: lfs@seva.com.
VHDL training, consulting services. Contact: Larry Saunders.

System Simulation Solutions
1100 NW Compton Drive, Beaverton, OR 97006, USA.
Phone: 503-690-1200. VHDL modeling services. Contact: James B. Morris.

Topdown Design Solutions
71 Spit Brook Road, Suite 301, Nashua, NH 03060, USA.
Phone: 603-888-8811, fax: 603-888-7694, email: sales@topdown.com. VHDL training courses and consulting services are also available. Contact: Art Pisani.

TM Associates Training Specialists
14420 S. Kelmsley Drive Oregon City, OR 97045, Phone: 503-656-4457, fax: 503-656-4775, email: twille@europa.com HDL training specialists. Contact: Tom Wille

TransGate Technologies
19310 Oak Street, Aloha OR 97007-4422 USA. Phone: 503-591-1232, email: ptkwt@teleport.com. Model development with emphasis on communications and application-specific memories, ASIC/PLD design. Contact: Phillip Tomson.

VAutomation Inc.
20 Trafalgar Square, Suite 443, Nashua, NH 03063 USA. Phone: 603-882-2282; fax: 603-882-1587; email: eric@vautomation.com; internet: http://www.vautomation.com Sythesizable HDL models. Contact: Eric Ryherd.

VHDL Technology Group
100 Broadhead Road, Suite 140, Bethlehem, PA 18017 USA.
Phone: 610-882-3130, fax: 610-882-3133, email: wdb@vhdl.com. VHDL training and consulting services. Contact: William D. Billowitch.

Widman Associates
70 Crestmont Drive, Oakland, CA 94619 USA. Phone: 510-482-3564,
fax 510-482-5339. Contact: Bob Widman.

Willamettte HDL, Inc.
14314 SW Allen Blvd. Suite 625, Beaverton, OR 97005 USA. Phone: 503-590-8499, fax 503-645-9728, email: info@whdl.com, internet: http://www.whdl.com. Top-down design methodologies for system, board, and ASIC design.

Online resources

email:
omf_info@cfi.org

newsnet news groups:
sci.electronics.cad
comp.arch.fpga
comp.lang.tcl
comp.lang.verilog
comp.lang.vhdl
comp.lsi
comp.lsi.testing

home pages:
http://www.model.com
http://www.vhdl.org/vhdl_intl
http://www.e2w3.com
http://www.translogiccorp.com
http://www.translogic.nl

web references:
Tcl/Tk man pages: http://www.elf.org/tcltk-man-html/contents.htm
SunScript (Tcl information): http://sunscript.sun.com
Open Verilog International: http://www.ovi.org

Index

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A

Adding signals or nets to windows 51
Application window 22
Authorization codes 19
  entering your permanent code 20
  Model Technology direct sales 19
  Model Technology OEM sale 19
  protect your investment 17
  time-based code 17
  updates 20

B

Breakpoints
  editing 54
  examining values at 54
  setting 52, 89
  viewing 52

C

Change directory 79
Comment characters in VSIM commands 14
Compile
  confirming compiled design units 97
  Verilog designs 108
  VHDL design unit ordering 95
  VHDL designs 82
Continue (CONT) tool bar button 89
Conventions
  text and command syntax 14
cursors, see Wave window
Customizing ModelSim windows 72

D

Dataflow window 35
  adding a process to 36, 46
  adding a signal to 36, 48
  mouse actions 36
  tracing signals and nets 36
Debugging a VHDL design 127
Default simulation run length 99
describe ModelSim command 90
Design library, creating 80

E

Editing a list 43
Editing signals in the Wave window 65
Email
  Model Technology's email address 16
Errors
  finding in VHDL design 102
  viewing in Source window 102
examine ModelSim command 90

F

Files
  viewing source files 53
Force
  deposit 50
  drive 50
  freeze 50
  repeat every time interval 50
  second value 50
Force values 49
Formatting a list 41

G

Graphic interface 21–76

H

HDL items

defined 14
Home page
Model Technology's home-page URL 16

I
Initialization file, see Project file

K
Keyboard shortcuts 76

L
Library, creating 80
List window 37
change display radix 115
editing a list 43
formatting a list 41
mouse actions 39
restoring formats 44
save listing 91
saving the format of 43
status bar 38, 39
triggering options 40

M
Mapping the work library 94
Menu bar
in List window 38
in Wave window 62
ModelSim 24
menu bar 38
ModelSim commands
comments in commands 14
ModelSim menu bar 24

N
Nets
adding to windows 51

P
PostScript file
from waveform display 71
Process window 45
mouse actions 46
status bar 46
Processes
displaying values and pathnames of 59
Project file
creating new file 81

Q
Quitting the simulator 91

R
Radix
specifying 41
Restoring a List window configuration 44
Run
menu bar commands 87

S
Saving a List window configuration 43
Saving a listing 43
Saving ModelSim window settings 78
Saving the Wave window configuration 72
Shortcuts
keyboard 76
Signals
adding to windows 51
forcing values 49
in the List and Wave windows 99
selecting which types to view 48
triggering listings for 42, 43
Signals window 47
mouse actions 48
status bar 47
Simulating
saving results as a list 37

Simulation
changing default run length 99
error correction 102
mixed VHDL/Verilog 120
restart 101
Verilog 109
VHDL 79
Source file
selecting 53
Source window 52
customizing 54
status bar 53
Status bar 87
Stimulus, apply to a design 86
Structure window 53, 56
mouse actions 58
status bar 58
System requirements 18

T

Tracing HDL items with the Dataflow window 36
Transcript window 32
keyboard actions 33
menu bar 24
status bar 32
Tutorial setup 78

U

Updating the List window 44

V

Variables
changing values during simulation 103
Variables window
mouse actions 60
status bar 60
Verilog
viewing design in the Structure and Source windows 122
Verilog simulation 105
VHDL simulation 79
View Active Processes 45
View Processes In Region 45

W

Wave window 61
adding HDL items 65
analog formatting 68
cursors 70
cursors on workstation 125
formatting 65
literal formatting 68
logic formatting 69
menu bar 62
mouse actions 63
PostScript output 71
saving configuration 72
status bar 63
zooming 69
Windows
changing positions and sizes 85
colors 73
customizing 72
fonts 73
saving 72
saving configurations 74
Work library mapping 94
Writing a PostScript file 71

Z

Zooming in the Wave window 69

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z